Certificação Linux

Guia prático para a prova LPIC-1 102

Casa do Código | alura

JULIANO RAMOS

Sumário

- ISBN
- Agradecimentos
- Sobre o livro
- Sobre o autor
- Prefácio
- Tópico 105: Shells e scripts do shell
  - 1 Tópico 105.1 – Personalizar e trabalhar no ambiente shell
  - 2 Tópico 105.2 – Customização e criação de script
- Tópico 106: Interfaces de usuário e Desktops
  - 3 Tópico 106.1 – Instalar e configurar o X11
  - 4 Tópico 106.2 – Áreas de trabalho gráficas
  - 5 Tópico 106.3 – Acessibilidade
- Tópico 107: Tarefas administrativas
  - 6 Tópico 107.1 – Tarefas administrativas
  - 7 Tópico 107.2 – Automatizar e agendar tarefas administrativas de sistema
  - 8 Tópico 107.3 – Localização e internacionalização
- Tópico 108: Serviços essenciais do sistema
  - 9 Tópico 108.1 – Manutenção da data e hora do sistema
  - 10 Tópico 108.2 – Log do sistema
  - 11 Tópicos 108.3 e 108.4
- Tópico 109: Fundamentos de rede
  - 12 Tópico 109 – Fundamentos de rede
- Tópico 110: Segurança
  - 13 Tópico 110.1 - Tarefas administrativas de segurança
  - 14 Tópico 110.2 – Segurança do host
  - 15 Tópico 110.3 – Proteção de dados com criptografia
  - 16 Conclusões finais e próximos passos rumo à certificação

ISBN

Impresso: 978-85-5519-375-0

Digital: 978-85-5519-376-7

Dados Internacionais de Catalogação na Publicação (CIP)
(Câmara Brasileira do Livro, SP, Brasil)

Ramos, Juliano
Certificação Linux : guia prático para a prova LPIC-1 102 / Juliano Ramos. -- 1. ed. -- São Paulo : AOVS Sistemas de Informática, 2024.

Bibliografia.
ISBN 978-85-5519-375-0

1. Certificação profissional 2. Linguagem de programação (Computadores) 3. LINUX (Sistema operacional de computadores) I. Título.

24-202728 CDD-005.43

Índices para catálogo sistemático:

1. Linux : Sistema operacional : Computadores : Processamento de dados 005.43

Aline Graziele Benitez - Bibliotecária - CRB-1/3129

Caso você deseje submeter alguma errata ou sugestão, acesse http://erratas.casadocodigo.com.br.

Agradecimentos

Quero agradecer de todo o coração a Deus, o criador de tudo que existe, e ao seu filho, o meu companheiro de jornada, Jesus Cristo, que me deu a inteligência e a perseverança para produzir este material.

Aproveito para agradecer ao fundador do Linux, Linus Torvalds, e a todos os desenvolvedores atuais do Kernel Linux e das centenas de softwares de código aberto que utilizo no dia a dia, em especial aos amigos programadores do GNOME e do Fedora Linux. Não posso deixar de fora o agradecimento ao revisor Antonio Loureiro, que me auxiliou com paciência e competência nas melhorias de ortografia e de raciocínio ao longo deste livro.

Por último, mas não menos importante, quero agradecer às duas mulheres que hoje compõem minha vida, minhas fontes de inspiração diária: Giovanna Santos, minha filha, que me faz ver a vida com mais alegria e esperança a cada manhã, e minha centelha do universo, minha companheira, que realiza um trabalho louvável na Delegacia da Mulher. Sandrinha, você me fez ver o mundo de forma mais objetiva, a vida boa é, sem dúvida, a vida simples, e tenho orgulho de ser o companheiro de uma mulher com tão nobre profissão, que arrisca a vida e se dedica ao próximo todos os dias. Aproveito o momento, para deixar aqui meu pedido:

— *Sandrinha, quer se casar comigo?*

A resposta, trago para vocês no próximo livro. Até lá, vamos que vamos.

Sobre o livro

Este livro aborda a administração de sistemas Linux com foco na prova de certificação LPIC-1 102-500 do *Linux Professional Institute* (LPI).

Uma **pessoa administradora de sistemas** tem uma função de grande importância para qualquer empresa ou organização que dependa da tecnologia da informação para funcionar. Essa pessoa é responsável pela configuração, operação e manutenção dos sistemas de rede e de computador para garantir disponibilidade, segurança e desempenho. Além disso, um(a) administrador(a) de sistemas deve possuir conhecimentos aprofundados sobre sistemas operacionais, hardwares, softwares, bancos de dados e nuvem, o que permite adequar a tecnologia às necessidades da empresa e resolver quaisquer problemas que surjam.

A pessoa administradora de sistemas também fornece suporte técnico aos usuários, instala e atualiza recursos, realiza backups de dados e monitora a segurança da rede. Ser uma pessoa administradora de sistemas requer capacitação profissional de qualidade, bem como habilidades de comunicação, organização e trabalho em equipe. É uma profissão desafiadora e gratificante, que contribui para o sucesso e a inovação das empresas e organizações.

A certificação Linux LPIC-1 102-500 é uma das provas necessárias para obter o nível LPIC-1, que avalia as competências básicas de uma pessoa administradora de sistemas Linux. A prova aborda temas como shells do Linux e scripts do shell, interfaces de usuário e desktops, tarefas administrativas, serviços essenciais do sistema, fundamentos de rede e segurança. A prova tem 60 questões de múltipla escolha ou preenchimento de lacunas, que devem ser respondidas em 90 minutos. O exame custa em média US$ 165 e tem validade de 5 anos.

Para preparar você para a prova 102-500 do LPI, neste livro vamos estudar os objetivos desse exame, que são:

- Instalar e atualizar o sistema operacional e os pacotes de software necessários;
- Monitorar o desempenho, a disponibilidade e a segurança do sistema;
- Gerenciar os usuários, os grupos, as permissões e os recursos do sistema;
- Configurar e administrar os serviços de rede, como DNS, DHCP, FTP, SSH etc.;
- Realizar backups e restaurações dos dados e do sistema;
- Solucionar problemas e resolver incidentes relacionados ao sistema;
- Documentar e seguir as boas práticas e os padrões de qualidade do sistema.

A quem se destina este livro

Este livro se destina a qualquer pessoa que deseja obter uma certificação profissional em Linux. A LPI é a organização mais gabaritada em capacitar profissionais para atuarem como administradores(as) de sistemas no mundo todo. Para que você se torne um(a) profissional certificado LPI de nível Administrador, é preciso que você faça dois exames, conhecidos como LPIC-1 101-500 e LPIC-1 102-500. Abordaremos aqui todo o conhecimento necessário para o exame 102-500.

Ao unir conhecimento técnico e uma abordagem acessível, este livro tem o objetivo de capacitar leitores e leitoras de todos os níveis de familiaridade com o Linux a compreender e explorar esse sistema em constante evolução. Este livro é o resultado da paixão de professor em compartilhar *insights* valiosos sobre servidores Linux como um grande potencial transformador de vidas.

Convido você para embarcar nesta jornada para explorar o emocionante universo da tecnologia e refletir sobre seu impacto duradouro em nossas vidas.

Sobre o autor

Juliano Ramos, ou Prof. Juliano Ramos, como é mais conhecido nas mídias sociais, é um entusiasta apaixonado por tecnologia, inovação e sua interseção com a vida cotidiana. Com mais de duas décadas de experiência no campo da Tecnologia da Informação, em especial servidores Linux e certificação LPI, o professor tem dedicado sua carreira à exploração das fronteiras emergentes da revolução digital.

Com formação acadêmica em Pedagogia e em Engenharia da Computação, Juliano Ramos tem mergulhado fundo na busca de uma didática fácil e sem tecnicismos para preparar pessoas para a prova de certificação LPI.

Ao longo de sua jornada profissional, o professor desempenhou papéis fundamentais em projetos inovadores, como o programa *Telecentros*, no qual coordenou mais de 50 centros de inclusão digital nas periferias de São Paulo. Seu compromisso com a disseminação do conhecimento técnico levou-o a ministrar palestras em diversos locais, como a Campus Party, o Fórum Internacional Software Livre (FISL), centros acadêmicos, universidades e workshops em empresas, contribuindo de forma contínua para a disseminação do Linux e de ferramentas de código aberto.

Além de suas realizações profissionais, Ramos é um comunicador habilidoso e comprometido em tornar conceitos tecnológicos complexos acessíveis a todos. Em seu canal no YouTube (http://youtube.com/profjulianoramos), o professor compartilha todos os dias conhecimento de forma acessível a quem tiver interesse. Como ele diz em todos os seus vídeos: "*A vida é boa e, com Deus, tudo vai dar certo.*"

Prefácio

por Cícero Woshington

O livro *Certificação Linux: Guia prático para a prova LPIC-1 102* é um manual objetivo e direto para todos os profissionais que precisam ampliar suas oportunidades de carreira e demonstrar isso na forma de uma certificação LPI. O responsável por esta obra, Juliano Ramos, é um professor com décadas de experiência e um profissional de TI com uma jornada técnica que o torna perito no assunto. E como se isso não bastasse, Linux é algo que o professor Juliano carrega na pele, literalmente.

Com uma abordagem clara para entender todos os tópicos da prova LPIC-1 102-500, cada capítulo aborda os comandos e conceitos necessários para compreender cada item e preparar você para as questões da prova de certificação. O *Tópico 105: Shells e scripts do shell* trata de personalizar o ambiente shell e escrever scripts Bash para automação de tarefas. O *Tópico 106: Interfaces de usuário e desktops* envolve a instalação e configuração do X11 e a compreensão dos principais ambientes de desktop no Linux. O *Tópico 107: Tarefas administrativas* aborda a administração de contas de usuário, grupos e arquivos de sistema, automação e agendamento de tarefas administrativas, bem como localização e internacionalização. O *Tópico 108: Serviços essenciais do sistema* inclui a manutenção da data e hora do sistema, configuração do log do sistema, fundamentos do Mail Transfer Agent (MTA) e configuração de impressoras e impressão. O *Tópico 109: Fundamentos de rede* envolve o entendimento dos protocolos de internet, configuração persistente de rede, solução de problemas simples de rede e configuração de DNS no cliente. Por fim, o *Tópico 110: Segurança* engloba tarefas administrativas de segurança, segurança do *host* e proteção de dados com criptografia.

Com a certificação Linux, profissionais demonstram competência na configuração de servidores, resolução de problemas específicos do

ambiente e gerenciamento eficiente de sistemas complexos. Essa certificação não só atesta as habilidades técnicas do indivíduo, mas também aumenta sua atratividade no competitivo mercado de trabalho de TI, abrindo portas para avanços e oportunidades profissionais. E ninguém melhor que o professor Juliano para mostrar os melhores caminhos para estudar, evoluir profissionalmente e dominar este que é o sistema líder de mercado para servidores e um habilitador de tantas carreiras.

Aproveite a leitura e bons estudos!

— *Cícero Woshington, mestre em Engenharia de Software e engenheiro DevOps sênior*

Tópico 105: Shells e scripts do shell

Capítulo 1
Tópico 105.1 – Personalizar e trabalhar no ambiente shell

Certificação	LPIC-1
Versão	5.0
Tópico no exame	105.1 – Personalizar e trabalhar no ambiente shell
Objetivo	Os candidatos devem ser capazes de personalizar o ambiente shell para atender às necessidades dos usuários. Também devem ser capazes de modificar arquivos de configuração global e de configuração local de usuários.
Peso	4

O **shell** do Linux é um programa CLI (*Command Line Input*) que permite ao usuário interagir com o sistema operacional através de linhas de comando. O shell é importante porque facilita a execução de tarefas, a automação de processos e a comunicação entre o usuário e o kernel Linux.

Existem vários tipos de shell no Linux, como o *bash*, o *zsh*, o *ksh* e o *tcsh*, cada um com suas vantagens e desvantagens. O shell **bash** é, de longe, o mais comum em todas as distribuições de Linux.

De forma prática, conhecer a linha de comando do shell torna você uma pessoa administradora de sistema mais efetiva em diversas tarefas. Vamos supor que você precisa informar a sua diretoria sobre

o espaço em disco usado em um servidor de Linux. No modo gráfico – conhecido como GUI (*Graphical User Interface*) –, seja no Linux ou no Windows, você dependeria de abrir o gestor de arquivos, procurar pelas propriedades de disco e obter essa informação.

O grande problema nesse processo é que existem diversas interfaces gráficas para o Linux, como KDE, GNOME, XFCE, Cinnamon, entre outras – e mesmo no Microsoft Windows, as coisas estão sempre mudando de lugar.

No caso de um servidor Linux, para realizar essa tarefa, basta abrir o shell de comando e digitar `df -h` para ter o resultado a seguir. Observe a porcentagem de uso de disco na 5ª coluna.

```
# df -h
Sist. Arq.      Tam. Usado Disp. Uso% Montado em
devtmpfs        4,0M     0  4,0M   0% /dev
tmpfs           3,8G   80M  3,7G   3% /dev/shm
tmpfs           1,5G  2,1M  1,5G   1% /run
/dev/nvme0n1p3  237G   51G  184G  22% /
/dev/nvme0n1p3  237G   51G  184G  22% /home
tmpfs           3,8G   11M  3,8G   1% /tmp
/dev/nvme0n1p2  974M  298M  609M  33% /boot
/dev/nvme0n1p1  599M   18M  582M   3% /boot/efi
tmpfs           766M  3,9M  762M   1% /run/user/1000
```

E esse comando não muda, indiferente de você utilizar Debian, Ubuntu, Linux Mint, Fedora, Red Hat ou qualquer outra distribuição de Linux.

1.1 Abrindo um terminal

Quando abrimos um terminal de comandos na GUI, ele é considerado um shell do tipo **PTS** (*pseudo terminal slave*); quando executado em um console do sistema, ele é um shell **TTY** (*teletypewriter*). *No primeiro caso, estamos lidando com um*

emulador de terminal. Por serem parte das sessões gráficas, os emuladores de terminal, como o Gnome-terminal *ou o* Konsole*, são ricos em recursos e fáceis de usar, se comparados aos terminais baseados em texto.

Abrindo um terminal no ambiente gráfico GNOME

Para abrir o terminal no ambiente de desktop GNOME, você pode seguir uma destas opções:

1. Método direto: pressione a tecla "Super" (tecla do logotipo do Windows) para abrir o menu de atividades. Digite `terminal` na barra de pesquisa e pressione `Enter` para abrir uma nova janela do terminal.
2. Método alternativo: pressione as teclas de atalho `Ctrl + Alt + T` para abrir uma nova janela do terminal.
3. Usando o menu de aplicativos:
    - Clique no menu "Atividades" (ícone no canto superior esquerdo da tela);
    - Clique no ícone "Aplicativos" no canto inferior esquerdo do menu de atividades;
    - Navegue até a categoria "Utilitários" ou "Acessórios";
    - Procure por "Terminal" na lista de aplicativos e clique no ícone para abrir o terminal.

Independente do método escolhido, ao seguir essas etapas, você abrirá uma nova janela do terminal no ambiente de desktop GNOME.

Abrindo um terminal no ambiente gráfico KDE

Para abrir o terminal no ambiente gráfico KDE, você pode seguir uma destas opções:

1. Método direto: pressione o atalho `Alt + F2` para abrir o lançador de aplicativos e digite `Konsole` na caixa de pesquisa. Ao executar esse comando, uma nova janela do terminal (Konsole) será aberta.

2. Método alternativo: clique com o botão direito em qualquer área vazia na área de trabalho ou no painel (barra de tarefas). No menu de contexto que aparece, selecione "Abrir Terminal" ou "Konsole".
3. Usando o menu de aplicativos:
    - Clique no menu "K" no canto inferior esquerdo da tela;
    - Navegue até a categoria "Utilitários" ou "Acessórios";
    - Procure por "Konsole" na lista de aplicativos e clique no ícone para abrir o terminal (Konsole).

Independente do método escolhido, ao seguir essas etapas, você abrirá uma nova janela do terminal (Konsole) no ambiente de desktop KDE.

> Dica: na maioria das distribuições de Linux, você consegue acessar os ambientes TTY combinando as teclas `Ctrl + Alt + F1-F6`.

1.2 Os arquivos de inicialização global

Os **arquivos de inicialização global** no Linux são arquivos e diretórios que são executados durante o processo de inicialização do sistema operacional. Eles são responsáveis por configurar variáveis de ambiente, definir opções de inicialização, carregar módulos de kernel e executar outras tarefas necessárias para preparar o sistema para uso.

As configurações realizadas em arquivos ou diretórios globais alteram a inicialização de todos os usuários do sistema. Sempre que um usuário inicia uma nova sessão do seu shell, uma série de configurações são carregadas para essa sessão. Os arquivos apresentados na tabela a seguir são alguns dos que controlam essas configurações e são lidos na inicialização do sistema:

Arquivo	Descrição
`/etc/profile`	O arquivo `/etc/profile` é um arquivo de script do shell usado pelo Bash e por outros shells compatíveis com POSIX. Ele é executado sempre que um usuário faz login no sistema e fornece as configurações padrão do ambiente para o shell de login. O arquivo contém variáveis de ambiente, configurações do prompt do shell, configurações de PATH e outras opções de configuração. Administradores do sistema podem personalizar o arquivo `/etc/profile` para fornecer configurações de ambiente específicas para o sistema. Os usuários também podem personalizar suas próprias configurações de ambiente, adicionando configurações ao seu arquivo `/.bash_profile` ou `/.bashrc` .
`/etc/profile.d/`	O diretório `/etc/profile.d` é usado em sistemas Unix e Linux para armazenar scripts que são executados automaticamente durante o processo de inicialização. Para que a pessoa administradora não precise colocar todas as suas configurações personalizadas da inicialização do usuário no arquivo `/etc/profile` , ela pode criar scripts em `/etc/profile.d` que são lidos em ordem alfabética. Por exemplo, se você tiver um arquivo chamado `custom.sh` dentro de `/etc/profile.d` , ele será executado automaticamente quando um usuário fizer login no sistema.
`/etc/bash.bashrc`	O objetivo do arquivo `/etc/bash.bashrc` é fornecer configurações globais para todos os

Arquivo	Descrição
	usuários do sistema que usam o Bash como shell padrão. Ele permite que administradores do sistema configurem opções específicas do Bash, *aliases* de comando, variáveis de ambiente e outras personalizações que devem ser aplicadas a todos os usuários.

Vamos a um exemplo prático de uma configuração global. Abra um terminal de comandos com permissão de usuário administrador (*root*) e digite:

```
# nano /etc/profile
```

Adicione ao final desse arquivo a linha:

```
LINUX="Eu uso todos os dias"
```

Salve este arquivo com `Ctrl + o` e feche o editor com `Ctrl + x`. Faça um novo login em um TTY usando as teclas `Ctrl + Alt + F6` e verifique se existe uma variável disponível no seu ambiente chamada `LINUX` que tem como conteúdo "*Eu uso todos os dias*". Para isso, você pode usar o seguinte comando:

```
# echo $LINUX
Eu uso todos os dias
```

Vale lembrar que essa variável adicionada no arquivo `/etc/profile` pertence a todos os usuários do sistema.

1.3 Inicialização em nível local

No contexto do Linux, os arquivos de inicialização local são arquivos de configuração e scripts que são executados no contexto de um

usuário específico durante o processo de inicialização do sistema operacional ou quando esse usuário faz login.

Ao contrário dos arquivos de inicialização global, que são aplicados a todos os usuários do sistema, os arquivos de inicialização local são específicos de cada usuário e são usados para personalizar o ambiente de trabalho e configurar opções individuais.

Arquivo	Descrição
`~/.bash_profile`	Esse arquivo específico do Bash é usado para configurar o ambiente do usuário. Também pode ser usado para buscar e executar `~/.bash_login` e `~/.profile`.
`~/.bash_login`	Também específico ao Bash, esse arquivo só será executado se não houver um arquivo `~/.bash_profile`. Seu nome sugere que ele deve ser usado para executar comandos necessários no login.
`~/.profile`	Esse arquivo não é específico ao Bash e só é originado se o `~/.bash_profile` ou `~/.bash_login` existir, o que normalmente é o caso. Assim, a principal finalidade de `~/.profile` é conferir se um shell Bash está sendo executado e, se estiver, buscar `~/.bashrc` e executá-lo, caso exista. Ele normalmente define a variável PATH para que ela inclua o diretório privado `~/bin` do usuário, se ele existir.
`~/.bash_logout`	Se existir, esse arquivo específico do Bash faz algumas operações de limpeza ao sair do shell. Isso pode ser conveniente em certos casos, como nas sessões remotas.

Vamos a um exemplo prático de configuração local, parecido com o que fizemos na configuração global. Abra o arquivo de configuração local do seu usuário:

```
$ nano ~/.profile
```

Adicione uma variável ao final do arquivo:

```
UNIX="É a origem do Linux"
```

Salve com `Ctrl + o` e feche com `Ctrl + x`. Faça login novamente com seu usuário e verifique a variável UNIX criada:

```
# echo $UNIX
É a origem do Linux
```

> Observação: devido à ordem em que os arquivos são executados, os arquivos locais têm precedência sobre os globais.

1.4 Quantos shells estão sendo executados?

Para que você possa conferir quantos shells do Bash estão rodando em seu sistema, no seu terminal digite o comando abaixo:

```
# ps aux | grep bash
juliano  38295  0.0  0.0 224648  5120 pts/0  Ss  14:04  0:00 bash
juliano  39367  0.1  0.0 224692  5120 tty4   Ss+ 14:06  0:00 -bash
juliano  39871  0.0  0.0 222560  2304 pts/0  S+  14:07  0:00 grep --
color=auto bash
```

Observe que, no resultado do comando anterior, o usuário `juliano` possui um terminal de shell (`pts/0`) aberto e um terminal em modo texto (`tty4`).

1.5 Criando um pequeno script em shell

Um script em shell, ou *shell script*, é um conjunto de instruções escritas em uma linguagem específica de shell como o Bash. Os scripts são usados para automatizar tarefas do sistema operacional, executar comandos e controlar o fluxo de execução de um programa. Comumente atribuímos a extensão `.sh` no script, mas vale ressaltar que em sistemas do tipo Unix e Linux a extensão não

é algo obrigatório. Um exemplo simples de script que exibe uma mensagem na tela seria:

```
#!/bin/bash
# Isto é um comentário. A linha acima especifica o interpretador a ser usado.
echo "Olá, mundo!"
# Exibe a mensagem "Olá, mundo!" no terminal.
```

Para executar esse script, salve o mesmo exemplo, `meu_script.sh` ; em seguida, adicione a permissão de execução com o comando:

```
# chmod +x meu_script.sh
```

E agora execute-o:

```
./meu_script.sh
```

Outra forma de executar esse script é com o comando `source` :

```
# source script.sh
```

Aliás o comando `.` é um atalho para o comando `source` . Por exemplo:

```
# . script.sh
```

Ou chamando o script com o shell `bash` :

```
# bash script.sh
```

1.6 Alias

O shell permite a criação de apelidos de comandos que chamamos de ***alias***. Isso permite que você atribua um nome ou atalho a um comando ou sequência de comandos mais longos. Ao criar um *alias*, você pode substituir um comando longo ou complexo por um nome mais curto e de fácil memorização. Isso pode economizar tempo e

esforço, especialmente se você precisar executar o mesmo conjunto de comandos repetidamente.

Os *aliases* são definidos no shell e permanecem ativos durante a sessão atual. No entanto, eles não são salvos permanentemente e serão perdidos quando você encerrar a sessão. Para tornar os *aliases* persistentes, você pode adicioná-los ao seu arquivo de configuração do shell, como o `.bashrc`.

A sintaxe básica para definir um *alias* é a seguinte:

```
alias nome_do_alias='comando ou sequência de comandos'
```

Por exemplo, para criar um *alias* chamado `lsd` que executa o comando `ls` com as opções `-l` e `-d`, você pode usar o seguinte comando:

```
alias lsd='ls -l -d'
```

Depois de definir esse *alias*, você pode simplesmente digitar `lsd` no seu shell, e o comando `ls -l -d` será executado em seu lugar. Você também pode incluir variáveis ou argumentos nos *aliases*. Por exemplo:

```
alias gocd='cd ~/Documentos/projetos/Go'
```

Neste caso, o *alias* `gocd` vai direcionar você para o diretório `~/Documentos/projetos/Go` sempre que você digitar `gocd` no shell.

1.7 O uso de funções no shell

No shell, as funções são blocos de código que podem ser definidos e executados dentro de um script ou na linha de comando. As funções permitem agrupar um conjunto de comandos relacionados em uma

única unidade, tornando o código mais organizado, modular e reutilizável.

Ao definir uma função no shell, você pode atribuir um nome a ela e especificar um conjunto de comandos que serão executados quando a função for chamada.

No shell Bash, existem duas formas de definir uma função: com ou sem o uso de parênteses após o nome da função.

Quando você define uma função sem parênteses após o nome, ela é chamada de função "clássica" do Bash. Por exemplo:

```
minha_funcao {
    # Corpo da função
    echo "Esta é uma função sem parênteses"
}
```

Essa forma de definição de função é compatível com versões mais antigas do Bash e é amplamente suportada. No entanto, seu uso pode levar a ambiguidades em certos casos, especialmente ao lidar com recursão ou ao passar parâmetros.

Por outro lado, ao definir uma função com parênteses após o nome, você está usando a sintaxe recomendada e mais moderna do Bash. Por exemplo:

```
minha_funcao() {
    # Corpo da função
    echo "Esta é uma função com parênteses"
}
```

Essa forma de definição de função é mais clara e menos suscetível a ambiguidades. Ela também é mais flexível ao lidar com recursão e passagem de parâmetros.

Ambas as formas podem ser chamadas da mesma maneira, usando apenas o nome da função:

```
minha_funcao
```

Em resumo, a principal diferença entre uma função que termina com parênteses e uma que não termina é a sintaxe e a clareza. A forma com parênteses é considerada a sintaxe recomendada e é mais robusta em termos de recursão e passagem de parâmetros. No entanto, a forma sem parênteses ainda é amplamente suportada e pode ser usada em ambientes que requerem compatibilidade com versões mais antigas do Bash.

Vejamos um exemplo de função com comandos:

```
# function funcao1 {
> date;
> uptime;
> uname -a;
> echo "Fim da funcao";
>}
```

Agora, execute a função:

```
# funcao1
```

Outra forma de criar a função para executar comandos é:

```
# function funcao2 { date; uptime; }
```

Execute:

```
# funcao2
```

Existe ainda uma terceira forma de se utilizar:

```
# funcao3 () { uptime
```

As funções são úteis para encapsular tarefas repetitivas, criar funcionalidades específicas e modularizar o código em scripts shell. Elas ajudam a melhorar a legibilidade, a manutenção e a reutilização do código, permitindo que você organize seu script em blocos lógicos e específicos.

1.8 Array no Linux

Um **array** é uma estrutura de dados que permite armazenar e organizar um conjunto de elementos do mesmo tipo de forma sequencial. Cada elemento dentro de um array é acessado por meio de um índice, que representa sua posição na sequência.

Os arrays são utilizados em programação para armazenar coleções de dados relacionados e são uma ferramenta fundamental para lidar com conjuntos de valores de maneira eficiente. Eles permitem o acesso rápido aos elementos, a manipulação de dados em massa e a implementação de algoritmos complexos.

No Bash, você pode declarar um array atribuindo valores a uma variável especial. Por exemplo, para criar um array chamado `frutas` que contém várias frutas, você pode fazer o seguinte:

```
frutas=("maçã" "banana" "laranja" "uva")
```

Nesse exemplo, o array `frutas` é inicializado com quatro elementos: `maçã`, `banana`, `laranja` e `uva`.

Para acessar elementos específicos do array, você pode usar um índice numérico entre colchetes. O índice começa em 0 para o primeiro elemento. Por exemplo, para acessar a primeira fruta do array `frutas`, você pode fazer o seguinte:

```
echo ${frutas[0]}
```

Isso imprimirá `maçã` no console. Você pode substituir o número entre colchetes para acessar outros elementos do array.

Também é possível obter todos os elementos do array usando `${frutas[@]}` ou `${frutas[*]}`. Por exemplo, para percorrer todos os elementos do array `frutas` e imprimi-los, você pode usar um loop `for` da seguinte forma:

```
for fruta in "${frutas[@]}"; do
    echo $fruta
```

```
done
```

Isso imprimirá cada fruta do array em uma nova linha.

Você também pode adicionar elementos a um array existente usando a sintaxe de atribuição. Por exemplo, para adicionar `abacaxi` ao array `frutas`, você pode fazer o seguinte:

```
frutas+=("abacaxi")
```

Agora o array `frutas` terá cinco elementos, incluindo o `abacaxi` adicionado.

Os arrays no shell Bash podem ser úteis para armazenar e manipular conjuntos de dados, como no exemplo das frutas. Eles permitem acessar facilmente elementos individuais, percorrer o array e adicionar ou remover elementos conforme necessário.

Só para relembrar

No Linux, assim como em outros sistemas operacionais, existem as variáveis *locais* e as variáveis *globais*, que são dois tipos distintos de variáveis.

Uma **variável local** é declarada dentro de um escopo específico, como uma função ou um bloco de código delimitado por chaves `{ }`. Essas variáveis são visíveis apenas dentro do escopo onde foram declaradas e não podem ser acessadas ou modificadas de fora desse escopo. Quando o escopo em que uma variável local está definida é encerrado, a variável é destruída e o seu valor é perdido.

Por exemplo, considere o seguinte trecho de código em Bash:

```
funcao_exemplo() {
    local variavel_local="Este é um exemplo"
    echo $variavel_local
}
```

```
funcao_exemplo
echo $variavel_local
```

A variável `variavel_local` é declarada como uma variável local dentro da função `funcao_exemplo`. Ela só é visível e utilizável dentro dessa função. Quando tentamos acessar a variável fora da função, como na segunda linha após a chamada `funcao_exemplo`, ocorrerá um erro, pois a variável não está mais acessível.

Dentro do terminal do shell Bash, podemos declarar a variável simplesmente colocando seu nome e valor. Por exemplo:

```
# LINUX="Sim, eu uso e recomendo!"
```

Por outro lado, uma **variável global** é declarada fora de qualquer escopo específico, tornando-a visível e acessível em qualquer parte do código, incluindo funções e blocos de código. Variáveis globais podem ser usadas em todo o programa e seus valores podem ser alterados e acessados em diferentes partes do código.

No terminal do Bash você pode declarar uma variável global usando o comando `export`, por exemplo:

```
# export LINUX="Sim, eu uso e recomando!"
```

Você também pode exportar uma variável local para global a qualquer momento usando o `export`:

```
# export LINUX
```

No entanto, é importante ter cautela ao usar variáveis globais, pois elas podem levar a problemas de legibilidade, manutenção e potenciais conflitos entre partes diferentes do código que acessem e modifiquem a mesma variável global.

Em resumo, a diferença entre variáveis locais e globais no Linux (e em outros sistemas operacionais) reside no escopo de visibilidade e na disponibilidade das variáveis. As variáveis locais são restritas a um escopo específico e têm uma vida útil limitada, enquanto as

variáveis globais são acessíveis em todo o programa e mantêm seu valor ao longo do tempo.

1.9 Comando env

No Linux, o comando `env` é utilizado para executar um programa em um ambiente de shell específico. Ele permite definir variáveis de ambiente temporárias antes de executar um comando ou script, alterando o ambiente padrão do shell.

O comando `env` é especialmente útil quando você deseja fornecer valores diferentes para variáveis de ambiente enquanto executa um programa ou script específico, sem afetar o ambiente global do sistema.

A sintaxe básica do comando `env` é a seguinte:

```
env [VARIAVEL=VALOR]... [COMANDO [ARGUMENTOS]...]
```

Aqui está uma explicação dos elementos principais:

- `[VARIAVEL=VALOR]...` : são variáveis de ambiente opcionais que você pode definir antes de executar o comando. Cada variável é especificada no formato "VARIAVEL=VALOR";
- `[COMANDO [ARGUMENTOS]...]` : é o comando ou script que você deseja executar com o ambiente modificado. Pode ser seguido por argumentos adicionais, se necessário.

Vamos ver um exemplo de uso do comando `env` . Suponha que você tenha um script chamado `meu_script.sh` que requer uma variável de ambiente chamada `LINGUAGEM` definida como `Python` . Você pode executar esse script com o comando `env` da seguinte maneira:

```
env LINGUAGEM=Python ./meu_script.sh
```

Neste exemplo, o comando `env` define a variável de ambiente `LINGUAGEM` com o valor `Python` e, em seguida, executa o script `./meu_script.sh` com o ambiente modificado.

O comando `env` também pode ser útil ao escrever scripts ou programas que precisam ser portáveis, pois você pode garantir que as variáveis de ambiente necessárias estejam definidas independentemente do ambiente em que o script é executado. Vejamos o exemplo com o comando `ls` quando roda em um terminal preparado para mostrar cores diferentes para diretórios, links simbólicos e executáveis:

```
ls -l --color /usr/bin
```

Observe no seu terminal que as cores classificam os tipos de arquivos apresentados na lista. Agora vejamos como podemos evitar essa classificação rodando o mesmo comando em um ambiente diferente:

```
env LS_COLORS="di=0;0:ln=0;0:ex=0;0" ls -l --color /usr/bin
```

Nesse caso estamos desativando as cores dos diretórios, links simbólicos e executáveis na variável de ambiente que usamos para configurar as cores do comando `ls` . Isso prova que no ambiente que foi criado pelo comando `env` temos uma configuração diferente da do ambiente que normalmente utilizamos.

Em resumo, o comando `env` no Linux é usado para executar um programa ou script em um ambiente de shell específico, permitindo que você defina variáveis de ambiente temporárias antes da execução. Ele oferece flexibilidade para personalizar o ambiente de execução de um comando ou script sem afetar o ambiente global do sistema.

1.10 Comando unset

Quando você define variáveis de ambiente ou atributos para variáveis em um ambiente de shell, elas ficam disponíveis para o sistema e para os processos subsequentes. O comando `unset` permite remover essas variáveis, tornando-as indefinidas. A sintaxe básica do comando `unset` é :

```
unset [opção] [nome_da_variavel]
```

Exemplo: para remover a variável de nome "LINUX" do sistema, usamos:

```
unset LINUX
```

Já para remover o atributo da variável e não a variável em si, usamos:

```
unset -v LINUX
```

O comando `unset` remove a variável apenas da sessão atual do shell. Se você deseja que a variável seja removida permanentemente, normalmente é necessário remover a definição da variável no arquivo de perfil do shell (por exemplo, no arquivo `~/.bashrc` para o Bash).

Conclusão

O tópico 105.1 da certificação LPIC-1 102 é de extrema importância no exame, uma vez que possui um peso 4, o que indica que você terá quatro questões relacionadas a ele. Dedicar uma atenção especial a esse tópico é essencial e pode contribuir para uma boa pontuação no exame.

Uma maneira eficaz de se preparar é criar novos exemplos com arrays e funções, sem medo de experimentar diferentes comandos e scripts do Linux. Aqui estão alguns exemplos que você pode explorar.

Exemplo 1: Manipulação de um array de nomes usando loops e comandos do Linux

```
nomes=("João" "Maria" "Pedro" "Ana")

# Imprimir todos os nomes do array
for nome in "${nomes[@]}"; do
    echo $nome
done

# Adicionar um novo nome ao array
nomes+=("Carlos")

# Imprimir o tamanho do array
echo "Tamanho do array: ${#nomes[@]}"

# Remover o terceiro nome do array
unset nomes[2]
```

Exemplo 2: Criação de uma função para calcular a soma de elementos em um array de números

```
calcular_soma() {
    local soma=0
    for numero in "$@"; do
        soma=$((soma + numero))
    done
    echo "A soma dos números é: $soma"
}

numeros=(5 10 15 20 25)
calcular_soma "${numeros[@]}"
```

Além disso, você pode explorar outras funcionalidades, como a manipulação de permissões de arquivos e diretórios, configuração de cotas de disco para usuários específicos, personalização do esquema de diretório `/etc/skel`, entre outros aspectos relevantes abordados no tópico.

Lembre-se de praticar e experimentar diferentes cenários para aprimorar seu conhecimento e familiaridade com os comandos e recursos do Linux relacionados ao tópico 105.1. Isso o ajudará a se

sentir mais confiante durante o exame e a obter uma pontuação sólida nessa seção.

Capítulo 2
Tópico 105.2 – Customização e criação de script

Certificação	LPIC-1
Versão	5.0
Tópico no exame	105.2 – Editar e escrever scripts simples
Objetivo	Os candidatos devem ser capazes de editar scripts existentes ou escrever um novo script simples do Bash.
Peso	4

O tópico 105.2 da certificação LPIC-1 tem como objetivo certificar que a pessoa administradora possua conhecimento na criação e edição de pequenos scripts em shell Bash.

Existem várias razões pelas quais criar scripts no shell do Linux pode ser vantajoso:

- **Automação de tarefas:** com scripts, você pode automatizar tarefas repetitivas ou complexas. Em vez de executar comandos individualmente, um script pode agrupar várias etapas e executá-las de uma vez;
- **Otimização de rotinas:** scripts podem ajudar a otimizar processos, melhorando a eficiência e reduzindo erros. Você pode criar scripts personalizados para realizar sequências específicas de comandos e garantir que sejam executados corretamente;

- **Personalização do ambiente:** os scripts permitem personalizar seu ambiente de trabalho. Você pode criar scripts para configurar suas preferências, definir variáveis de ambiente, criar atalhos de teclado personalizados, entre outros;
- **Automação de instalação e configuração:** ao criar scripts de instalação e configuração, você pode automatizar a configuração inicial de um sistema Linux, instalar pacotes, configurar serviços e ajustar as configurações do sistema de acordo com suas necessidades;
- **Consistência e padronização:** usar scripts garante consistência e padronização nas tarefas executadas. Todos os membros da equipe podem executar os mesmos scripts para realizar as mesmas tarefas, garantindo resultados consistentes;
- **Facilidade de manutenção:** quando você precisa atualizar ou modificar um processo, é mais fácil fazer isso em um único script em vez de alterar vários comandos individuais. Isso facilita a manutenção e a atualização dos processos no futuro.

Em resumo, a criação de scripts no shell do Linux oferece flexibilidade, automação e personalização, tornando o gerenciamento do sistema mais eficiente e adaptado às suas necessidades específicas.

2.1 A estrutura básica de um script

Um arquivo de script é, basicamente, uma sequência ordenada de comandos que devem ser executados. A estrutura básica de um script consiste em:

- ***Shebang*:** o *shebang* é a primeira linha do script e especifica o interpretador a ser usado para executar o script. Em um script

Bash, o *shebang* padrão é `#!/bin/bash`. Isso informa ao sistema operacional que o script deve ser interpretado pelo Bash;

- **Comentários:** os comentários são linhas no script que são ignoradas pelo interpretador, mas fornecem informações úteis para os leitores do código. Em Bash, os comentários começam com o caractere `#`.

- **Declaração de variáveis:** você pode declarar variáveis para armazenar valores no script. Em Bash, não é necessário especificar explicitamente o tipo de variável. Basta atribuir um valor a uma variável usando o sinal de igual (`=`). Por exemplo, `nome='João';`

- **Execução de comandos:** você pode executar comandos no script Bash. Basta digitar o comando diretamente no script e ele será executado quando o script for executado. Por exemplo, "ls" para listar os arquivos em um diretório;

- **Estruturas de controle:** o Bash oferece várias estruturas de controle, como condicionais (`if` - `else`), loops (`for` , `while`) e casos (`case`), que permitem controlar o fluxo de execução do script com base em condições ou iterar sobre um conjunto de valores;

- **Funções:** você pode definir funções no script para agrupar um conjunto de comandos relacionados. Isso permite reutilizar o código em diferentes partes do script ou em scripts diferentes;

- **Entrada e saída:** o script pode interagir com o usuário por meio da entrada (por exemplo, lendo valores fornecidos pelo usuário) e da saída (por exemplo, exibindo mensagens na tela). O comando `read` pode ser usado para ler a entrada do usuário, enquanto o comando `echo` pode ser usado para exibir mensagens.

Essa é apenas uma estrutura básica e existem muitos recursos avançados disponíveis no shell Bash. No entanto, esses elementos fornecem uma base sólida para a criação de scripts em Bash e o que

é necessário para o tópico 102.5. Agora que compreendemos um pouco do esqueleto de um script, prepare seu cafezinho, vamos começar a programar.

2.2 Criando um pequeno script

Vamos começar com um exemplo simples para compreendermos melhor a estrutura de um script. Salve este conteúdo como `script-simples.sh`:

```
#!/bin/bash

# Comentário: Este é um script de exemplo

# Declaração de variáveis
nome="João"
idade=30

# Exibição de uma mensagem
echo "Olá, $nome! Você tem $idade anos."

# Execução de um comando
data=$(date +%d-%m-%Y)
echo "Hoje é $data."

# Estrutura condicional
if [ $idade -gt 18 ]; then
  echo "Você é maior de idade."
else
  echo "Você é menor de idade."
fi

# Função
function exibir_mensagem {
  echo "Esta é uma função de exemplo."
}
```

```
# Chamada da função
exibir_mensagem

# Fim do script
```

Vamos destrinchar esse script:

- O script começa com o shebang `#!/bin/bash`, seguido por comentários que fornecem informações sobre o script. Em seguida, temos a declaração de variáveis, onde `nome` recebe o valor `"João"` e `idade` recebe o valor `30`.
- Depois disso, utilizamos o comando `echo` para exibir mensagens na tela, interpolando as variáveis usando `$nome` e `$idade`;
- Em seguida, usamos o comando `date` para obter a data atual e armazenamos o resultado na variável `data`;
- Após isso, temos uma estrutura condicional `if-else` que verifica se a `idade` é *maior que 18* e exibe a mensagem correspondente;
- Por fim, definimos uma função chamada `exibir_mensagem` e a chamamos usando `exibir_mensagem`.

Como já vimos no capítulo anterior, para executar esse script, é necessário conceder a ele permissão de execução com o comando `chmod +x script-simples.sh`; depois, execute-o com `bash script-simples.sh`.

2.3 Utilizando parâmetros no script

Assim como usamos parâmetros nos comandos do Linux (por exemplo, `ls -lha` ou `tar -xvzf`), podemos colocar parâmetros em nosso script. Vamos começar com um exemplo simples:

```
#!/bin/bash
clear
```

```
echo "------------------Meu primeiro script--------------------"
echo "O meu script se chama $0"
echo "Esse script recebeu $# parâmetros, que são $1 $2"
echo ""
uptime
free -h
df -h
whoami
echo "-----------------Fim do script-------------------------"
```

Quando executar esse script, coloque alguns parâmetros fictícios, por exemplo:

```
bash script.sh -t -s
```

Destrinchando:

- `#!/bin/bash` : essa linha é o *shebang*, que especifica o interpretador a ser usado para executar o script. Nesse caso, indica que o interpretador é o Bash;
- `clear` : usado para limpar a tela do terminal, deixando-a vazia antes de exibir a saída do script;
- `echo "------------------Meu primeiro script--------------------"` : exibe uma mensagem na tela informando "Meu primeiro script". O texto entre aspas é exatamente o que será exibido na tela;
- `echo "O meu script se chama $0"` : exibe outra mensagem na tela mostrando o nome do script. O `$0` é uma variável especial que contém o nome do próprio script;
- `echo "Esse script recebeu $# parâmetros que são, $1 $2"` : exibe uma mensagem mostrando o número de parâmetros que foram passados para o script (`$#`) e quais são esses parâmetros (`$1` e `$2`). O `$#` é uma variável especial que indica o número de parâmetros passados, e `$1` e `$2` são as variáveis que armazenam o primeiro e o segundo parâmetro, respectivamente.
- `echo ""` : exibe uma linha vazia, apenas para dar uma separação visual entre as informações exibidas;

- `uptime` : exibe informações sobre o tempo de atividade do sistema, mostrando quanto tempo o sistema está ligado e a carga média do sistema.
- `free -h` : exibe informações sobre o uso de memória do sistema, mostrando a quantidade de memória livre e utilizada;
- `df -h` : exibe informações sobre o uso de espaço em disco, mostrando o espaço livre e utilizado em cada sistema de arquivos montado no sistema;
- `whoami` : exibe o nome do usuário atualmente logado no sistema;
- `echo "------------------Fim do script-------------------------"` : exibe uma mensagem na tela, indicando que o script chegou ao fim.

Esse script básico exibe algumas informações do sistema, como o tempo de atividade, uso de memória e espaço em disco, além de mostrar o nome do script e os parâmetros passados para ele. É um exemplo simples para ilustrar como o Bash pode ser usado para exibir informações e realizar comandos no terminal.

Para tornar o script mais completo, vamos criar uma interação com o usuário. Para isso, usaremos o comando `read` , que permite solicitar entrada do usuário e armazenar essa entrada em uma variável:

```
#!/bin/bash
clear
echo "-------------------Meu primeiro script--------------------"
echo "O meu script se chama $0"
echo "Esse script recebeu $# parâmetros, que são $1 $2"
echo "Por favor digite um valor"
read VAR1
echo "Seu valor foi $VAR1"

echo "-------------------Fim do script-------------------"
```

Quando executar esse script, você terá que informar um valor, que será armazenado na variável `VAR1` .

2.4 Comando de seleção ou tomada de decisão

Para deixar um script mais robusto, podemos usar comandos de tomada de decisão. Para exemplificar, vamos pensar em um jogo de videogame: o personagem, para abrir um determinado baú, precisa encontrar uma certa chave. Se ele tem a chave, o baú abre; se ele não tem, o baú não abre. Observe a condição ***se*** utilizada no exemplo: essa condição é muito comum em praticamente todas as linguagens de programação, incluindo o shell script. Vejamos como ela funciona no shell Bash:

```
if [condição]
then
Ações
fi
```

Onde:

- **Condição:** teste que, se verdadeiro, passará o controle para o bloco dentro do `then` ;
- **Ações:** comandos a serem executados se o resultado da **condição** for verdadeiro.

É muito comum a pessoa que está desenvolvendo esquecer de fechar o `if` . Lembre-se sempre de que, para cada `if` aberto, você precisa fechá-lo com o comando `fi` , que é `if` ao contrário.

Antes de continuarmos com um exemplo prático, vamos conhecer uma lista dos comandos mais utilizados para se testar uma condicional:

Comando	Descrição
valor1 = valor2	Significa que valor1 é igual a valor2.

Comando	Descrição
valor1 != valor2	Significa que valor1 difere de valor2.
valor1 -eq valor2	Significa que valor1 é igual a valor2.
valor1 -ne valor2	Significa que valor1 não é igual a valor2.
valor1 -gt valor2	Significa que valor1 é maior que valor2.
valor1 -ge valor2	Significa que valor1 é maior ou igual a valor2.
valor1 -lt valor2	Significa que valor1 é menor que valor2.
valor1 -le valor2	Significa que valor1 é menor ou igual a valor2.

Outras opções de uso comum dentro de testes condicionais são:

- `-e nome_do_arquivo` : verifica se o arquivo existe;
- `-d nome_do_arquivo` : verifica se o arquivo é um diretório;
- `-f nome_do_arquivo` : verifica se o `nome_do_arquivo` é um arquivo comum.

Exemplo prático:

```
#!/bin/bash
echo "Digite um número qualquer:"
read numero;

if[ "$numero" -gt 20 ];
then
echo "Este número é maior que 20!"
fi
```

2.5 O comando else

Existe a possibilidade de tratarmos a tomada de decisão quando ocorre falha. Para isso, usamos o comando `else`, cuja sintaxe é:

```
if [condição]
then
Ações_1
else
Ações_2
fi
```

Onde:

- `condição` : teste que, se verdadeiro, passará o controle para o bloco dentro de `then` ;
- `Ações_1` : comandos a serem executados se o resultado de **condição** for verdadeiro;
- `Ações_2` : comandos a serem executados se o resultado de **condição** for falso.

No exemplo a seguir, verificamos se um número digitado pelo usuário é positivo ou negativo:

```
#!/bin/bash
echo "Digite um número:"
read numero;
if ["$numero" -ge 0]
then
    echo "O número $numero é positivo"

else
    echo "O número $numero é zero ou negativo"
fi
```

2.6 O comando elif

Há casos em que temos mais de uma condição a ser testada, todas correlacionadas. Para isso, usamos o comando `elif`, cuja sintaxe é:

```
if [ condição_1 ];
then
    Ações_1

elif [ condição_2 ];
then
    Ações_2

elif [ condição_3 ];
then
    Ações_3

    fi
```

Vamos criar um exemplo prático de sua utilização:

```
#!/bin/bash
echo "Selecione uma opção:"
echo "1 - Exibir a hora do sistema:"
echo "2 - Mostrar o diretório atual:"
echo "3 - Exibir uma mensagem qualquer:"
read opcao;

if [ $opcao == "1"];
then
data=`date`
echo "$data"

elif [ $opcao == "2"];
then
diretorio=`pwd`
echo "$diretorio"

elif [ $opcao == "3"];
```

```
then
echo "Digite seu nome:"
read nome;
echo "Bem-vindo $nome";
fi
```

2.7 O comando case

O comando `case` no shell Bash é uma estrutura de controle que permite realizar comparações múltiplas em uma variável ou valor. Ele é frequentemente usado quando há várias opções possíveis a serem testadas em uma única variável, e é uma opção mais visual em relação ao `if` e ao `elif`.

A sintaxe básica do comando `case` é a seguinte:

```
case variável in
  padrão1)
    comandos1 ;;
  padrão2)
    comandos2 ;;
  padrão3)
    comandos3 ;;
  *)
    comandos_padrão ;;
esac
```

Vamos analisar cada parte dessa estrutura:

- `variável` : é a variável que será testada em diferentes padrões. Pode ser uma variável numérica, de cadeia de caracteres (string) ou outro tipo suportado pelo Bash;
- `padrão1` , `padrão2` , `padrão3` etc.: são os padrões que serão testados. Cada padrão é comparado com o valor da variável;
- `comandos1` , `comandos2` , `comandos3` etc.: são os comandos a serem executados se o padrão correspondente for verdadeiro;

- `*)` : o asterisco é um padrão curinga que corresponde a qualquer valor que não tenha sido correspondido pelos padrões anteriores;
- `comandos_padrão` : são os comandos a serem executados caso nenhum dos padrões anteriores tenha sido correspondido.
- Após a palavra-chave `esac` (*case* escrito ao contrário), o bloco do comando `case` é encerrado.

Quando o comando `case` é executado, ele avalia a variável em cada padrão na ordem em que são listados. Se um padrão corresponder ao valor da variável, os comandos associados a esse padrão serão executados. O comando `;;` é usado para indicar o fim dos comandos de cada padrão.

A estrutura do comando `case` é útil quando você tem várias opções possíveis para verificar em uma variável e precisa executar diferentes ações com base nesses valores correspondidos. É uma forma eficiente e legível de lidar com várias condições em um único bloco de código.

Usando o mesmo exemplo do comando `elif` , vamos agora montar o script com o `case` :

```
#!/bin/bash
echo "Selecione uma opção:"
echo "1 - Exibir a hora do sistema:"
echo "2 - Mostrar o diretório atual:"
echo "3 - Exibir uma mensagem qualquer:"
read opcao;

case $opcao in
"1")
data=`date`
echo $date
;;

"2")
diretorio=`pwd`
```

```
echo $diretorio
;;

"3")
echo "Informe o seu nome:"
read nome;
echo "Bem-vindo $nome"
;;
esac
```

Destrinchando, o script em shell fornecido é um exemplo de utilização do comando `case` para criar um menu de opções interativo. Vamos analisar o código linha por linha:

```
#!/bin/bash
```

Essa linha é o `shebang`, que indica o interpretador a ser usado para executar o script — neste caso, o Bash.

```
echo "Selecione uma opção:"
echo "1 - Exibir a hora do sistema:"
echo "2 - Mostrar o diretório atual:"
echo "3 - Exibir uma mensagem qualquer:"
```

Essas linhas usam o comando `echo` para exibir as opções disponíveis ao usuário. São mostradas três opções, cada uma com uma descrição.

```
read opcao;
```

Essa linha solicita ao usuário que digite sua opção e armazena o valor na variável `opcao` por meio do comando `read`.

```
case $opcao in
"1")
data=`date`
echo $date
;;
```

Aqui começa o bloco `case`, que avalia o valor da variável `opcao`. Se o valor for igual a `"1"`, o bloco de comandos abaixo do padrão `"1"` é

executado. No caso, o comando `date` é executado para obter a data e hora do sistema e, em seguida, o valor obtido é exibido com o comando `echo`.

```
"2")
diretorio=`pwd`
echo $diretorio
;;
```

Se o valor de `opcao` for igual a `"2"`, o bloco de comandos abaixo do padrão `"2"` é executado. Aqui, o comando `pwd` é executado para obter o diretório atual em que o script está sendo executado, e em seguida, o valor é exibido com o comando `echo`.

```
"3")
echo "Informe o seu nome:"
read nome;
echo "Bem-vindo $nome"
;;
```

Se o valor de `opcao` for igual a `"3"`, o bloco de comandos abaixo do padrão `"3"` é executado. O usuário é solicitado a digitar seu nome por meio do comando `read`, e então uma mensagem de boas-vindas é exibida, interpolando o nome fornecido com o comando `echo`.

```
esac
```

Essa linha marca o final do bloco `case`.

2.8 Loops condicionais

O comando `for` é uma estrutura de controle de fluxo utilizada para executar um bloco de código repetidamente com base em uma lista de valores. No Bash, o `for` é frequentemente usado em conjunto com o comando `seq`, utilizado para gerar uma sequência de números.

A sintaxe básica do comando `seq` é a seguinte:

```
seq [opções] VALOR_INICIAL VALOR_FINAL
```

Aqui estão algumas opções comuns do comando `seq`:

- `-w` ou `--equal-width`: garante que todos os números tenham a mesma largura, preenchendo com zeros à esquerda, se necessário;
- `-s` ou `--separator`: define o separador entre os números da sequência;
- `-f` ou `--format`: permite especificar um formato personalizado para a saída.

Agora vamos ver um exemplo de uso do comando `for` com o `seq` para percorrer uma sequência de números:

```
for i in $(seq 1 5); do
    echo "Número: $i"
done
```

Nesse exemplo, o comando `seq 1 5` gera uma sequência de números de 1 a 5. O `for` então percorre cada número da sequência, atribuindo-o à variável `i`. Em seguida, o bloco de código dentro do `for` é executado, que, neste caso, imprime o valor atual de `i` utilizando o comando `echo`.

A saída desse código seria:

```
Número: 1
Número: 2
Número: 3
Número: 4
Número: 5
```

O `for` executou o bloco de código cinco vezes, uma vez para cada número na sequência gerada pelo `seq`. O comando `for` com `seq` pode ser útil em várias situações, especialmente quando você

precisa executar um conjunto de comandos repetidamente com base em uma sequência numérica.

Vamos a um exemplo prático. Suponha que você tenha uma série de arquivos com nomes semelhantes, como `arquivo1.txt`, `arquivo2.txt`, até `arquivo10.txt`, e deseje renomeá-los. Você pode usar o comando `for` em conjunto com `seq` para percorrer a sequência de números e renomear os arquivos correspondentes. Por exemplo:

```
for i in $(seq 1 10); do
    mv arquivo$i.txt novo_arquivo$i.txt
done
```

Esse código renomeia os arquivos de `arquivo1.txt` a `arquivo10.txt` para `novo_arquivo1.txt` a `novo_arquivo10.txt`.

2.9 Loop while

Enquanto o loop `for` é ideal para quando sabemos até quando contar, o loop `while` é bom para quando não temos essa noção, mas sabemos de uma condição que deverá ser atendida para que o laço termine. Por exemplo:

```
#!/bin/bash
echo "Informe o que desejar, -1 para sair"
read dado;
while [ $dado != "-1"];
do
echo "Você digitou $dado"
read dado;
done
```

No exemplo, o usuário poderá digitar qualquer coisa, que o programa só fechará quando ele digitar `-1`.

A sintaxe básica do comando `while` é a seguinte:

```
while condição; do
    # bloco de código a ser executado
done
```

A condição é uma expressão que é avaliada antes de cada iteração do loop. Se a condição for verdadeira, o bloco de código dentro do `while` é executado. Após a execução do bloco, a condição é verificada novamente, e o processo se repete até que a condição se torne falsa. Quando a condição se torna falsa, o loop é encerrado e a execução continua após o `done`.

Aqui está um outro exemplo simples para ilustrar o uso do comando `while`:

```
contador=1
while [ $contador -le 5 ]; do
    echo "Contador: $contador"
    contador=$((contador + 1))
done
```

Nesse exemplo, o loop `while` é executado enquanto o valor da variável `contador` for *menor ou igual* a `5`. Dentro do bloco de código, é exibido o valor atual do contador usando o comando `echo`. Em seguida, o valor do contador é incrementado em 1 usando a expressão `$((contador + 1))`. Isso é necessário para garantir que a condição do loop eventualmente se torne falsa e o loop termine.

A saída desse código seria:

```
Contador: 1
Contador: 2
Contador: 3
Contador: 4
Contador: 5
```

Esse é apenas um exemplo simples para ilustrar o uso do comando `while`. No mundo real, você pode ter condições mais complexas e realizar várias operações dentro do bloco de código. O comando

`while` é útil para quando você precisar repetir um conjunto de comandos enquanto uma condição for verdadeira.

Um exemplo prático do uso do comando `while` seria em um script que faz o monitoramento de um serviço ou processo específico. O script pode executar repetidamente um bloco de código enquanto o serviço estiver em execução e interromper o loop quando o serviço for encerrado.

Aqui está um exemplo simplificado:

```
#!/bin/bash

# Iniciar o serviço
iniciar_servico

# Verificar se o serviço está em execução
while pgrep -f "nome_do_servico" >/dev/null; do
    # Realizar ações enquanto o serviço estiver em execução
    echo "O serviço está em execução..."
    sleep 5
done

# O serviço foi encerrado
echo "O serviço foi encerrado."
```

Nesse exemplo, o script começa iniciando o serviço por meio da função `iniciar_servico`. Em seguida, entra em um loop `while` que verifica se o serviço está em execução usando o comando `pgrep`. Enquanto o serviço estiver em execução (ou seja, o comando `pgrep` retornar um código de saída `0`), o bloco de código dentro do `while` é executado.

Dentro do bloco de código, você pode realizar ações adicionais, como monitorar o desempenho do serviço, enviar notificações ou tomar decisões com base no estado do serviço. Nesse exemplo, o bloco de código simplesmente exibe a mensagem "O serviço está em execução..." e pausa por 5 segundos usando o comando `sleep`.

Quando o serviço for encerrado (ou seja, quando o comando `pgrep` não encontrar o processo correspondente ao serviço), a condição do `while` se tornará falsa e o loop será interrompido. Em seguida, o script exibirá a mensagem "O serviço foi encerrado.".

Esse é apenas um exemplo básico de uso do comando `while`. Na prática, você pode personalizar o bloco de código dentro do `while` para atender às suas necessidades específicas, realizando ações mais complexas enquanto aguarda a condição de saída ser atendida.

Conclusão

Neste capítulo, exploramos de forma prática como criar scripts eficientes utilizando o shell Bash. A certificação atribui um peso significativo de 4 a esse conteúdo, destacando sua importância.

É comum eu afirmar nas empresas em que atuo que um profissional de TI realmente habilidoso é, essencialmente, preguiçoso. Acredite ou não, isso faz todo o sentido! Ao contrário do profissional mediano, que se vê constantemente envolvido em situações de emergência e vivencia um alto nível de estresse, o bom profissional prefere se dedicar ao estudo do problema em questão e criar automações para solucioná-lo.

Portanto, evite gastar seu dia inteiro executando tarefas repetitivas. Automatize suas atividades, crie scripts e desfrute de uma rotina mais feliz e produtiva. É nesse momento que se compreende o verdadeiro poder dos scripts em shell!

Tópico 106: Interfaces de usuário e Desktops

Capítulo 3
Tópico 106.1 – Instalar e configurar o X11

Certificação	LPIC-1
Versão	5.0
Tópico no exame	106.1 – Instalar e configurar o X11
Objetivo	Os candidatos devem ser capazes de instalar e configurar o X11.
Peso	2

O sistema de janelas utilizado em sistemas operacionais baseado em Unix é o **X Window System**, que foi desenvolvido no MIT (Instituto de Tecnologia de Massachusetts) na década de 1980. Ele funciona em um modelo cliente-servidor, onde o **servidor X** é responsável por controlar o hardware gráfico e gerenciar a exibição das janelas, enquanto os aplicativos clientes enviam comandos para o servidor para desenhar elementos gráficos na tela.

O X Window System é altamente configurável e extensível. Ele permite que os usuários personalizem a aparência do ambiente de desktop, alterando temas, esquemas de cores, fontes e outros elementos visuais. Além disso, ele suporta extensões, que são módulos de software complementares que fornecem recursos adicionais, como suporte a transparência, efeitos visuais avançados e manipulação de dispositivos de entrada especiais.

Uma das características distintivas do X Window System é o uso do protocolo de rede X11. Isso significa que as comunicações entre o servidor X e os aplicativos clientes ocorrem por meio da rede, permitindo o compartilhamento de recursos gráficos em um ambiente de rede.

Nos sistemas Linux modernos, como o Ubuntu e o Fedora, o X Window System ainda é amplamente utilizado, embora esteja sendo gradualmente substituído pelo Wayland, um protocolo de exibição mais recente e moderno. No entanto, o X ainda é suportado e amplamente utilizado devido à sua estabilidade, compatibilidade com uma vasta gama de aplicativos e recursos avançados de personalização e extensibilidade.

Cada instância de um servidor X em execução possui um nome de exibição que a identifica. Esse nome contém o seguinte:

```
hostname:displaynumber.screennumber
```

O `hostname` refere-se ao nome do sistema que exibirá o aplicativo. Se o nome de exibição não contiver o nome do hospedeiro, o host local será pressuposto.

O `displaynumber` faz referência à coleção de telas que estão em uso, seja uma única tela de laptop ou diversas telas em uma estação de trabalho. Cada sessão do servidor X em execução recebe um número de exibição começando em 0. Ao adicionar uma segunda tela, a referência será: 0.0 para a primeira tela, 0.1 para a segunda tela e assim por diante. O `screennumber` padrão é 0, que significa apenas uma tela física.

O nome de exibição de sua sessão X é armazenado na variável `DISPLAY`:

```
$ echo $DISPLAY
:0
```

Para iniciar um aplicativo em uma tela específica, atribua o número da tela à variável de ambiente `DISPLAY` antes de iniciar o aplicativo.

Por exemplo:

```
DISPLAY=:0.1 firefox &
```

No exemplo do comando acima, estamos abrindo o Firefox na segunda tela do computador.

3.1 Configurando o X

Atualmente, o diretório de configuração padrão do servidor X está localizado em `/etc/X11` . No entanto, é importante destacar que, nos dias de hoje, é cada vez menos comum precisarmos realizar configurações manualmente nesse servidor. A maioria das distribuições Linux automatizou o processo de configuração para o mouse, monitor e teclado, eliminando a necessidade de um arquivo de configuração específico para esses dispositivos. No entanto, em alguns casos específicos, como o uso de uma placa de vídeo não convencional, pode ser necessário ajustar manualmente o arquivo de configuração do servidor X.

É importante ressaltar que, para a maioria dos usuários, essas configurações manuais não são mais necessárias, pois as distribuições Linux se esforçam para fornecer uma experiência *plug-and-play*, facilitando a detecção e configuração automática dos dispositivos de entrada e saída.

O primeiro passo para gerar um arquivo de configuração do X é parar o modo gráfico. Você faz isso com o seguinte comando:

```
# systemctl isolate multi-user.target
```

Este comando vai parar o servidor X e você somente poderá usar a linha de comando. Para gerar o arquivo de configuração, que será necessário para alguma configuração manual, execute:

```
# Xorg --configure
```

O arquivo vai ser gerado dentro de `/root` com o nome `xorg.conf.new`. Para que os ajustes realizados nesse arquivo tenham efeito no sistema, você deve copiá-lo para o diretório `/etc/X11/` com o nome `Xorg.conf`.

```
# cp /root/Xorg.conf.new /etc/X11/Xorg.conf
```

Exemplo de um arquivo `Xorg.conf`:

```
Section "ServerLayout"
    Identifier     "Layout0"
    Screen      0  "Screen0" 0 0
    InputDevice    "Keyboard0" "CoreKeyboard"
    InputDevice    "Mouse0" "CorePointer"
EndSection

Section "Files"
EndSection

Section "Module"
    Load "glx"
EndSection

Section "InputDevice"
    Identifier     "Keyboard0"
    Driver         "kbd"
EndSection

Section "InputDevice"
    Identifier     "Mouse0"
    Driver         "mouse"
    Option         "Protocol" "auto"
    Option         "Device" "/dev/input/mice"
    Option         "ZAxisMapping" "4 5 6 7"
EndSection

Section "Monitor"
    Identifier     "Monitor0"
    VendorName     "Monitor Vendor"
    ModelName      "Monitor Model"
```

```
    HorizSync       30.0 - 83.0
    VertRefresh     50.0 - 76.0
EndSection

Section "Device"
    Identifier     "Device0"
    Driver         "nvidia"
    VendorName     "NVIDIA Corporation"
    BoardName      "GeForce GTX 1080"
    Option         "RegistryDwords" "PowerMizerEnable=0x1;
PerfLevelSrc=0x2222; PowerMizerDefault=0x3"
EndSection

Section "Screen"
    Identifier     "Screen0"
    Device         "Device0"
    Monitor        "Monitor0"
    DefaultDepth    24
    SubSection     "Display"
        Depth       24
        Modes      "1920x1080"
    EndSubSection
EndSection
```

Vamos analisar algumas seções e opções importantes do arquivo:

- `ServerLayout` : essa seção define a organização geral do servidor X. Nesse exemplo, há um único layout identificado como `Layout0` , que possui uma tela, um teclado e um mouse associados;

- `InputDevice` : as seções `InputDevice` são responsáveis por configurar os dispositivos de entrada, como o teclado e o mouse. No exemplo, temos as seções `"Keyboard0"` e `"Mouse0"` , onde são especificados os drivers correspondentes e algumas opções de configuração;

- `Monitor` : a seção `Monitor` define as configurações do monitor. Aqui é possível especificar o identificador, o nome do fabricante,

o modelo, as faixas de frequência horizontal e vertical suportadas pelo monitor;

- `Device` : a seção `Device` configura o driver de vídeo e o hardware relacionado. Nesse exemplo, estamos usando o driver `"nvidia"` para uma placa de vídeo GeForce GTX 1080. Além disso, há uma opção personalizada `"RegistryDwords"` que define algumas configurações específicas da placa de vídeo;

- `Screen` : a seção `Screen` especifica as configurações relacionadas à tela, como o dispositivo associado (`Device`), o monitor a ser utilizado (`Monitor`), a profundidade de cor padrão (`DefaultDepth`) e as resoluções suportadas (`Modes`).

Essas são apenas algumas das seções e opções mais comuns encontradas no arquivo `Xorg.conf` . É importante lembrar que as configurações podem variar dependendo do hardware e das necessidades específicas do sistema. É recomendado consultar a documentação relevante e recursos adicionais para obter informações mais detalhadas.

3.2 Os logs

O arquivo `~/.xsession-errors` é um arquivo de log que registra mensagens de erro e informações relacionadas à sessão X (servidor gráfico) de um usuário específico. Ele está localizado no diretório pessoal do usuário, representado pelo `~` (til) no Linux e outros sistemas do tipo Unix.

Quando você faz login em um ambiente gráfico como o GNOME ou o KDE, o servidor X inicia e cria uma sessão para o seu usuário. Durante essa sessão, se ocorrerem erros relacionados à exibição gráfica, à inicialização de aplicativos ou a outros componentes do ambiente gráfico, essas mensagens de erro serão registradas no arquivo `~/.xsession-errors` .

Aqui estão algumas informações adicionais sobre o arquivo `~/.xsession-errors`:

- **Propósito**: o arquivo `~/.xsession-errors` é útil para diagnosticar problemas relacionados à sessão X. Ele pode conter informações sobre erros, avisos, mensagens de depuração e outros registros relevantes para a sessão gráfica;
- **Formato**: o arquivo `~/.xsession-errors` é um arquivo de texto simples. Cada linha do arquivo geralmente corresponde a uma mensagem específica, que indica o horário e a natureza do registro;
- **Tamanho e rotação**: o arquivo `~/.xsession-errors` pode crescer ao longo do tempo, especialmente se houver muitos erros ou mensagens de depuração sendo registrados. Em algumas distribuições Linux, o arquivo pode ser truncado ou rotacionado periodicamente para evitar que se torne muito grande;
- **Acesso e permissões**: o arquivo `~/.xsession-errors` pertence ao usuário específico e tem permissões de leitura para esse usuário. Somente o próprio usuário ou o superusuário (*root*) têm acesso ao arquivo;
- **Leitura e análise**: para ler o conteúdo do arquivo `~/.xsession-errors`, você pode usar um editor de texto simples ou comandos como `cat` ou `less`. Por exemplo: `cat ~/.xsession-errors`.

O comando `cat` no Linux é usado para concatenar e exibir o conteúdo de arquivos. O nome "cat" é derivado de "concatenate", que sugere sua função principal de combinar arquivos. No entanto, o `cat` é frequentemente utilizado para exibir o conteúdo de um único arquivo no terminal.

Já o comando `less` é usado no terminal do Linux para visualizar o conteúdo de arquivos de texto de uma maneira paginada e interativa. Ele permite que você role para frente e para trás pelo

arquivo, procure por palavras-chave e execute outras operações de navegação. O `less` é especialmente útil para visualizar arquivos grandes sem carregar todo o conteúdo de uma vez, o que pode economizar recursos do sistema.

É importante mencionar que as mensagens de erro registradas nesse arquivo podem variar de acordo com a distribuição Linux, o ambiente gráfico e os aplicativos específicos que estão sendo usados. Portanto, ao analisar o `~/.xsession-errors` , é essencial considerar as informações contextuais relevantes para solucionar problemas específicos relacionados à sessão X.

3.3 Acesso ao servidor X

O acesso ao servidor X é controlado pelo comando `xhost` . Ele permite que você controle a lista de hosts que podem ter acesso a aplicações gráficas no servidor X.

Suponha que você queira permitir que o host com o endereço IP 192.168.1.100 se conecte ao seu servidor X. Você pode usar o seguinte comando:

```
xhost +192.168.1.100
```

Isso permitirá que o host com o endereço IP especificado acesse seu servidor X e exiba aplicativos gráficos. Se você desejar revogar essa permissão de acesso posteriormente, pode usar o seguinte comando:

```
xhost -192.168.1.100
```

Isso negará o acesso do host com o endereço IP 192.168.1.100 ao servidor X. Tenha atenção ao especificar o endereço IP do host ao qual você deseja permitir ou negar o acesso.

3.4 O diretório xorg.conf.d

O diretório `xorg.conf.d` contém arquivos de configuração modularizados que são usados para configurar diferentes aspectos do servidor X. Anteriormente, as configurações do Xorg eram frequentemente mantidas em um único arquivo de configuração chamado `xorg.conf`. No entanto, a abordagem modular do diretório `xorg.conf.d` permite uma maior flexibilidade e facilita a organização das configurações.

A estrutura do diretório `xorg.conf.d` é projetada para permitir que várias configurações sejam separadas em arquivos individuais. Cada arquivo contém configurações específicas para determinados dispositivos, drivers ou configurações globais. Esses arquivos são processados em uma ordem alfanumérica, o que permite um controle preciso sobre a prioridade das configurações.

Por exemplo, um sistema pode ter arquivos de configuração específicos para o touchpad, teclado, placas gráficas, entre outros, cada um em seu próprio arquivo no diretório `xorg.conf.d`.

Aqui está um exemplo hipotético de como os arquivos no diretório `xorg.conf.d` podem ser organizados:

```
/etc/X11/xorg.conf.d/
|-- 10-keyboard.conf
|-- 20-mouse.conf
|-- 30-touchpad.conf
|-- 40-graphics.conf
|-- 50-monitor.conf
|-- 60-screen.conf
|-- 70-input.conf
|-- 80-extensions.conf
|-- 90-serverflags.conf
```

Cada arquivo numerado indica a ordem em que as configurações serão aplicadas. Por exemplo, `10-keyboard.conf` pode conter

configurações específicas para o teclado, e `40-graphics.conf` pode conter configurações relacionadas à placa gráfica.

Os administradores do sistema podem adicionar, remover ou modificar arquivos nesse diretório para ajustar a configuração do servidor X de acordo com os requisitos específicos do hardware ou preferências do usuário. Isso oferece uma maneira flexível e organizada de lidar com as configurações gráficas.

Conclusão

O tópico 106.1 da LPIC-1 aborda o gerenciamento do servidor gráfico X. Neste capítulo, foram discutidos conceitos importantes como o arquivo de configuração `Xorg.conf` e seu papel na personalização e ajuste das configurações do servidor X.

Vimos que, atualmente, as distribuições de Linux automatizaram a configuração do mouse, monitor e teclado, o que elimina a necessidade de configuração manual por meio do arquivo `Xorg.conf`. No entanto, em casos específicos, como o uso de placas de vídeo diferentes, pode ser necessário ajustar manualmente esse arquivo.

Foi abordado o arquivo `~/.xsession-errors`, que registra informações e erros relacionados à sessão X. Esse arquivo pode ser útil para solucionar problemas e identificar falhas durante a inicialização do servidor gráfico.

Também foi mencionado o comando `xhost`, que é utilizado para controlar as permissões de acesso ao servidor X. Ele possibilita permitir ou negar o acesso de hosts específicos ao servidor gráfico.

Em resumo, o tópico 106.1 da LPIC-1 fornece informações básicas sobre o gerenciamento do servidor gráfico X no Linux, abordando o arquivo de configuração `Xorg.conf`, o arquivo `~/.xsession-errors` e o comando `xhost`. Conhecer esses conceitos é útil para pessoas administradoras de sistemas Linux que desejam personalizar e solucionar problemas relacionados ao servidor X.

CAPÍTULO 4
Tópico 106.2 – Áreas de trabalho gráficas

Certificação	LPIC-1
Versão	5.0
Tópico no exame	106.2 – Desktops gráficos
Objetivo	Os candidatos devem conhecer os principais desktops do Linux. Além disso, os candidatos devem conhecer os protocolos utilizados para acessar sessões de desktop remoto.
Peso	1

A GUI (*Graphical User Interface*, ou Interface Gráfica do Usuário) é responsável por fornecer elementos visuais, como janelas, ícones, menus e botões, que tornam a interação com o computador mais fácil e intuitiva. No Linux, o X Window System (abordado no capítulo anterior) fornece os recursos básicos, mas a experiência completa do usuário depende dos componentes oferecidos pelo ambiente de área de trabalho.

Um dos componentes mais importantes de um ambiente de área de trabalho é o *gerenciador de janelas*, que controla o posicionamento e a aparência das janelas. Diferentemente do Windows, que possui um único gerenciador de janelas, o Linux oferece várias opções, sendo as mais conhecidas o KDE, o GNOME, o XFCE e o Cinnamon. Esses gerenciadores de janelas apresentam interfaces gráficas distintas, estilos visuais e conjuntos de recursos exclusivos, para que usuários personalizem sua experiência de acordo com suas preferências.

Além dos gerenciadores de janelas, os ambientes de área de trabalho do Linux também incluem outros componentes, como painéis, lançadores de aplicativos, widgets e configurações de temas. Esses elementos permitem aos usuários acessar rapidamente aplicativos, ajustar configurações, visualizar informações úteis e personalizar a aparência e a funcionalidade do ambiente de área de trabalho de acordo com suas necessidades.

No geral, a GUI do Linux oferece uma variedade de opções e recursos para tornar a interação com o sistema mais agradável e produtiva. Os diferentes ambientes de área de trabalho fornecem interfaces personalizáveis e intuitivas, que permitem que usuários adaptem o sistema operacional às suas preferências individuais.

O exame de certificação não cobra conhecimento específico de um ambiente desktop, já que é possível realizar toda a administração do sistema através da linha de comando. A seguir, teremos uma descrição básica dos ambientes citados.

4.1 KDE

O KDE (K Desktop Environment) é um ambiente de área de trabalho livre e de código aberto projetado para sistemas operacionais baseados em Unix, como o Linux, FreeBSD e outros. É conhecido por fornecer uma interface gráfica completa e altamente personalizável. O ambiente possui uma variedade de aplicativos integrados que abrangem uma ampla gama de funcionalidades, como gerenciador de arquivos (Dolphin), navegador web (Konqueror), suíte de escritório (Calligra), reprodutor de mídia (Elisa), entre outros. Esses aplicativos são projetados para trabalhar de forma harmoniosa dentro do ambiente KDE, proporcionando uma experiência de usuário consistente.

Com o KDE, você tem a liberdade de escolher entre uma ampla variedade de temas para personalizar a aparência do ambiente de acordo com suas preferências. Além disso, você pode alterar os ícones utilizados nos aplicativos e pastas, tornando a experiência visualmente única.

Os painéis no KDE são altamente configuráveis, permitindo que você os posicione em diferentes partes da tela, adicione ou remova widgets e ajuste o tamanho deles. Isso oferece uma flexibilidade excepcional para criar um layout de área de trabalho que atenda às suas necessidades específicas.

O KDE também oferece uma ampla gama de widgets para a área de trabalho, como relógios, calendários, monitores de sistema, previsão do tempo e muito mais. Esses widgets podem ser adicionados e personalizados de acordo com suas preferências, fornecendo acesso rápido a informações relevantes e funcionalidades adicionais.

Em suma, o poder de personalização do KDE é notável, permitindo que os usuários modifiquem profundamente o visual e a configuração do ambiente de área de trabalho. Embora possa parecer complexo inicialmente, essa flexibilidade oferece a oportunidade de criar uma experiência verdadeiramente personalizada e adaptada às suas preferências individuais.

4.2 GNOME

O GNOME é um ambiente de área de trabalho popular para sistemas operacionais baseados em Unix, como o Linux. Ele é conhecido por sua simplicidade, elegância e usabilidade, oferecendo uma experiência de usuário intuitiva e moderna.Algumas características distintas do GNOME incluem:

- **Design minimalista:** o GNOME possui um design minimalista e limpo, com uma interface de usuário simplificada. Ele se

concentra em apresentar apenas as informações e controles necessários para oferecer uma experiência de uso sem distrações;
- **Atalhos e gestos:** o GNOME faz uso extensivo de atalhos de teclado e gestos de toque para agilizar as tarefas comuns. Isso permite que os usuários naveguem rapidamente entre aplicativos, alternem entre áreas de trabalho virtuais e acessem recursos importantes com facilidade;

- **Barra superior e atividades:** o GNOME possui uma barra superior que exibe informações do sistema, como hora, data e notificações. Nela, usuários podem acessar o menu de atividades, que exibe uma visão geral de todos os aplicativos em execução, áreas de trabalho virtuais e recursos adicionais, como pesquisa e configurações;

- **Experiência centrada no teclado:** o GNOME valoriza a experiência de uso do teclado, permitindo que os usuários naveguem e interajam com o ambiente sem o uso do mouse. Isso é útil para usuários que preferem usar atalhos de teclado e agilizar suas tarefas diárias;

- **Integração com serviços on-line:** o GNOME oferece uma integração fácil com serviços online populares, como Google Drive e Nextcloud. Isso permite que os usuários acessem e gerenciem seus arquivos e dados armazenados na nuvem diretamente do ambiente de área de trabalho;

- **Extensões do GNOME Shell:** o GNOME Shell, o componente principal do GNOME, pode ser ampliado por meio de extensões. Essas extensões permitem adicionar funcionalidades extras, personalizar a aparência e adaptar o ambiente às preferências individuais dos usuários.

O GNOME é apreciado por sua abordagem minimalista, foco na usabilidade e suporte a recursos modernos. Ele oferece uma experiência de área de trabalho fluida e intuitiva, atendendo às

necessidades de usuários que valorizam a simplicidade e a elegância em sua interação com o sistema operacional.

4.3 XFCE

O XFCE é um ambiente de área de trabalho leve e de código aberto projetado para sistemas operacionais baseados em Unix, como o Linux. Ele se destaca por sua eficiência em termos de recursos do sistema, sendo uma escolha popular para computadores mais antigos ou com especificações de hardware mais limitadas. Algumas características distintas do XFCE incluem:

- **Leveza e desempenho:** o XFCE é conhecido por sua eficiência em termos de uso de recursos do sistema. Ele consome menos memória e poder de processamento em comparação com outros ambientes de área de trabalho mais pesados, o que o torna ideal para dispositivos com recursos limitados ou para usuários que preferem um ambiente de área de trabalho ágil e rápido;
- **Interface intuitiva:** o XFCE possui uma interface intuitiva e fácil de usar. Ele segue um layout tradicional de menu no estilo de "Iniciar" (*Start Menu*) e uma barra de tarefas na parte inferior da tela. Isso torna a navegação pelos aplicativos e o gerenciamento de janelas simples e diretos;
- **Personalização:** apesar de ser um ambiente de área de trabalho leve, o XFCE oferece opções de personalização significativas, apesar de usuários ainda terem a possibilidade de alterar a aparência, temas, ícones e até mesmo o layout da área de trabalho para se adequarem às suas preferências individuais;
- **Gerenciamento de janelas:** o XFCE inclui um gerenciador de janelas chamado *Xfwm*, que permite controlar o posicionamento e o redimensionamento das janelas. Ele oferece recursos úteis, como agrupamento de janelas, atalhos de teclado

personalizáveis e gerenciamento eficiente de múltiplas áreas de trabalho virtuais;
- **Extensibilidade:** o XFCE pode ser estendido por meio de plug-ins e complementos. Existem várias extensões disponíveis que adicionam recursos extras, como relógios adicionais, lançadores de aplicativos, monitores de sistema e muito mais. Isso permite aos usuários personalizar ainda mais seu ambiente de área de trabalho e adaptá-lo às suas necessidades específicas.

O XFCE é apreciado por sua simplicidade, leveza e facilidade de uso. Ele oferece uma experiência de área de trabalho estável e funcional, adequada tanto para usuários iniciantes quanto para usuários experientes que valorizam um ambiente de trabalho eficiente e com baixo consumo de recursos.

4.4 Cinnamon

O ambiente Cinnamon foi desenvolvido pela equipe da distribuição de Linux, chamada de: `Linux Mint` como uma alternativa ao ambiente de desktop GNOME Shell, que, na época, passou por mudanças significativas em sua interface de usuário. Aqui estão algumas características e aspectos do ambiente Cinnamon:

- **Interface familiar e intuitiva:** Cinnamon foi projetado para proporcionar uma experiência de usuário familiar, especialmente para aqueles que estão acostumados com ambientes de desktop tradicionais, como o GNOME 2. Ele tem um layout de desktop clássico, incluindo um painel no fundo da tela e um menu de aplicativos.

- **Painel personalizável:** o painel do Cinnamon é altamente personalizável. Os usuários podem adicionar applets, como relógio, indicadores de sistema, entre outros, e também podem ajustar a aparência do painel de acordo com suas preferências.

- **Menu de aplicativos:** o menu de aplicativos do Cinnamon é projetado para ser simples e fácil de navegar. Ele organiza os aplicativos em categorias e fornece uma barra de pesquisa para facilitar a localização de programas instalados.
- **Configurações do sistema:** Cinnamon oferece uma variedade de opções de configuração do sistema por meio de seu painel de controle. Isso permite que os usuários personalizem facilmente a aparência, comportamento e configurações do sistema de acordo com suas preferências.
- **Efeitos visuais:** embora seja projetado para ser leve e responsivo, o Cinnamon inclui alguns efeitos visuais suaves, como animações de janelas, transições de desktop e outros elementos visuais que contribuem para uma experiência agradável.
- **Comunidade ativa:** o Cinnamon é apoiado por uma comunidade ativa de desenvolvedores e usuários. Novas versões são regularmente lançadas com melhorias e correções de bugs.
- **Disponibilidade em outras distribuições:** embora seja mais comumente associado ao Linux Mint, o Cinnamon pode ser instalado em outras distribuições Linux. Muitas distribuições oferecem o Cinnamon como uma opção de ambiente de desktop durante a instalação do sistema operacional.

4.5 Acesso remoto

As sessões de desktop remoto são suportadas nativamente pelo X, que emprega o *X Display Manager Control Protocol* (XDMCP) para se comunicar com monitores remotos. O XDMCP foi projetado para ser usado em redes locais (LANs) e não é recomendado para uso em

redes públicas devido a preocupações com segurança. Ele não fornece criptografia de dados, o que significa que as informações transmitidas pelo protocolo podem ser interceptadas ou manipuladas por terceiros.

Devido às preocupações de segurança e ao desenvolvimento de tecnologias mais avançadas, como o SSH (Secure Shell), o uso do XDMCP diminuiu ao longo dos anos. No entanto, em ambientes de rede controlados e seguros, o XDMCP ainda pode ser usado para fornecer acesso remoto a sessões gráficas em servidores do tipo Unix.

O *Virtual Network Computing* (VNC) é uma ferramenta multiplataforma que permite visualizar e controlar desktops remotos usando o protocolo *Remote Frame Buffer* (RFB). Ele permite que os eventos do teclado e mouse locais sejam transmitidos para o ambiente de trabalho remoto, enquanto as atualizações de tela são enviadas de volta para exibição localmente.

Vários servidores VNC podem ser executados na mesma máquina; cada um requer uma porta TCP exclusiva na interface de rede para aceitar solicitações de sessão de entrada. Por convenção, o primeiro servidor VNC usa a porta TCP 5900, o segundo usa 5901 e assim por diante.

O servidor VNC não exige privilégios especiais para ser executado, permitindo que um usuário comum inicie seu próprio servidor VNC a partir de sua conta remota. Em seguida, qualquer aplicativo cliente VNC pode ser usado na máquina local para acessar o desktop remoto, desde que as portas de rede correspondentes estejam acessíveis.

O arquivo `~/.vnc/xstartup` é um script de shell executado pelo servidor VNC durante a inicialização e pode ser usado para definir o ambiente de desktop disponibilizado para o cliente VNC.

É importante observar que o VNC não oferece criptografia ou autenticação nativas, portanto, é recomendado usar aplicativos de

terceiros que forneçam esses recursos. Frequentemente, são utilizados métodos que envolvem túneis VPN e SSH para proteger as conexões VNC.

O *Remote Desktop Protocol* (RDP) é usado principalmente para acessar desktops de sistemas operacionais Microsoft Windows através da porta TCP 3389. Embora o RDP utilize o protocolo proprietário da Microsoft, a implementação do cliente nos sistemas Linux consiste em programas de código aberto licenciados sob a GNU *General Public License* (GPL) e não possui restrições legais de uso.

O *Simple Protocol for Independent Computing Environments* (Spice) é um conjunto de ferramentas usado para acessar ambientes de desktop em sistemas virtualizados, tanto localmente quanto remotamente. Além disso, o protocolo Spice oferece recursos nativos para integração entre sistemas locais e remotos, como acesso a dispositivos locais (por exemplo, alto-falantes e dispositivos USB conectados) da máquina remota, além do compartilhamento de arquivos entre os dois sistemas.

Conclusão

Este é um tópico relativamente tranquilo no exame LPIC-1 102. Como ele tem peso 1, provavelmente haverá apenas uma pergunta teórica sobre um dos protocolos mencionados (RDP, VNC, XDMCP ou Spice). O conteúdo deste capítulo é suficiente para este tópico do exame.

Capítulo 5
Tópico 106.3 – Acessibilidade

Certificação	LPIC-1
Versão	5.0
Tópico no exame	106.3 – Acessibilidade
Objetivo	Demonstrar conhecimento e saber que existem tecnologias de acessibilidade.
Peso	1

O ambiente de desktop do Linux inclui muitas configurações e ferramentas para adaptar a interface do usuário a pessoas com deficiência. Os dispositivos de interface humana comuns — tela, teclado e mouse/touchpad — podem ser reconfigurados individualmente para compensar deficiências visuais ou uma mobilidade reduzida.

É possível, por exemplo, adaptar o esquema de cores da área de trabalho para melhor atender pessoas daltônicas. Além disso, quem sofre de lesões por esforço repetitivo pode tirar proveito de métodos alternativos de digitação e clique.

Alguns desses recursos de acessibilidade são fornecidos pelo próprio ambiente de área de trabalho, como o GNOME ou o KDE, e outros são fornecidos por programas adicionais. É importante escolher a ferramenta que melhor se integra ao ambiente de trabalho, para ter um auxílio de melhor qualidade.

5.1 Configurações de acessibilidade

As principais distribuições de Linux oferecem recursos de acessibilidade semelhantes, que podem ser personalizados por meio de um módulo especial no gerenciador de configurações do ambiente de desktop. No GNOME, esse módulo é chamado de "Acesso Universal", enquanto no KDE ele está localizado em "Configurações do Sistema", na seção de "Personalização" e "Acessibilidade".

Aqui estão alguns exemplos de recursos de acessibilidade comuns encontrados nas principais distribuições de Linux:

- **Leitores de tela**: os leitores de tela são programas que fornecem uma saída de áudio ou Braille para ajudar pessoas com deficiência visual a interagir com o computador. O Orca é um exemplo popular de leitor de tela no Linux;
- **Ampliação de tela**: essa funcionalidade permite ampliar partes específicas da tela para facilitar a leitura. O Compiz e o GNOME Shell oferecem recursos de ampliação de tela;
- **Teclado virtual**: um teclado virtual exibe uma representação visual de um teclado na tela, permitindo que usuários com dificuldades motoras ou limitações de mobilidade possam digitar usando o mouse ou outro dispositivo de entrada alternativo;
- **Controle por voz**: os recursos de controle por voz permitem que os usuários realizem tarefas no computador usando comandos de voz. O KDE, por exemplo, oferece suporte ao reconhecimento de voz através do seu módulo "Acessibilidade";
- **Alto contraste**: a opção de alto contraste ajusta o esquema de cores da interface do usuário para torná-la mais visível para pessoas com deficiência visual. Geralmente, é possível encontrar essa opção nas configurações de acessibilidade do ambiente de desktop;
- **Legendas e suporte a Libras**: alguns ambientes de desktop e reprodutores de mídia no Linux fornecem recursos para exibir

legendas de vídeos e suporte para a língua de sinais (Libras e outras), tornando o conteúdo multimídia mais acessível para pessoas com deficiência auditiva;
- **Personalização de atalhos de teclado**: usuários podem personalizar os atalhos de teclado para adaptá-los às suas necessidades individuais. Isso é particularmente útil para pessoas com deficiências motoras, que podem ter dificuldade em usar determinadas combinações de teclas.

Esses são apenas alguns exemplos de recursos de acessibilidade disponíveis no Linux. Cada distribuição pode ter suas próprias implementações e opções de personalização. É importante destacar que a acessibilidade no Linux é um esforço contínuo, e novos recursos e aprimoramentos são adicionados regularmente para melhorar a experiência de uso para todos os usuários.

Conclusão

O exame LPIC 102 contém apenas uma questão referente ao conteúdo sobre acessibilidade visto neste capítulo, portanto não requer um conhecimento aprofundado sobre o assunto. No entanto, é importante ressaltar que existem ferramentas de acessibilidade de código aberto disponíveis no Linux, como o leitor de tela Orca. Atualmente, os desktops Linux têm a capacidade de oferecer às pessoas com necessidades especiais recursos comparáveis aos sistemas proprietários. É um avanço significativo e que possibilita uma experiência acessível e inclusiva para todos os usuários, independentemente de suas necessidades específicas.

Tópico 107: Tarefas administrativas

Capítulo 6
Tópico 107.1 – Tarefas administrativas

Certificação	LPIC-1
Versão	5.0
Tópico no exame	107.1 – Administrar contas de usuário, grupos e arquivos de sistema relacionados
Objetivo	Os candidatos devem ser capazes de adicionar, remover, suspender e modificar contas de usuários.
Peso	5

O Linux é um sistema multiusuário e, portanto, possui um esquema de permissões que garante a privacidade e/ou compartilhamento de arquivos entre os usuários. Os tipos de usuários no Linux são:

- **Usuário administrador (superusuário):** o usuário conhecido como "*root*" no sistema é o administrador do sistema e tem controle total sobre todas as funcionalidades. No entanto, é importante ter cautela ao utilizá-lo, pois qualquer ação equivocada pode resultar em danos ao sistema;
- **Usuários do sistema:** esses usuários são responsáveis pelo controle de serviços e não precisam fazer login no sistema. Eles não devem ter senhas definidas e não possuem um shell válido para login. Um exemplo de usuário do sistema é o "*www-data*", usado exclusivamente para controlar o servidor web Apache;

- **Usuários comuns:** esses usuários são criados para aqueles que operam ou trabalham com o sistema Linux. Cada usuário comum ou administrador deve ter sua própria conta separada e, em geral, é recomendável evitar o uso da conta "*root*" para tarefas cotidianas. A utilização de uma conta de usuário comum proporciona uma camada adicional de segurança, pois as ações realizadas por essa conta são restritas aos seus próprios arquivos e recursos.

É importante seguir as boas práticas de segurança ao administrar um sistema Linux, atribuindo privilégios adequados a cada usuário e limitando o uso da conta "*root*" para tarefas que exigem acesso administrativo específico. Dessa forma, é possível garantir a segurança e a integridade do sistema.

6.1 O login

É altamente recomendável adotar uma política de senhas forte tanto para o usuário *root* quanto para os usuários comuns, a fim de evitar que invasores se aproveitem de senhas fracas. Para fazer login no sistema Linux, são necessários cinco elementos:

- **Nome de usuário:** é o identificador único atribuído a cada usuário no sistema. Ele é usado para identificar o usuário durante o processo de login;

- **Senha:** a senha é uma sequência de caracteres que serve como uma forma de autenticação para verificar a identidade do usuário. É importante escolher senhas fortes, combinando letras maiúsculas e minúsculas, números e caracteres especiais, a fim de aumentar a segurança;

- **Diretório *home*:** o diretório *home* é o local onde os arquivos e configurações pessoais de cada usuário são armazenados. Ele fornece um ambiente personalizado para o usuário, incluindo

suas configurações de shell, arquivos pessoais e outros dados relacionados;
- **Shell:** o shell é um programa que fornece uma interface de linha de comando para interagir com o sistema operacional. Ele permite que o usuário execute comandos e execute programas no sistema. Existem vários shells disponíveis no Linux, como o *bash*, *zsh*, entre outros;

- **Grupo primário:** cada usuário é atribuído a um grupo primário, que é um conjunto de usuários com permissões e acessos semelhantes. O grupo primário define as permissões padrões do usuário em relação aos arquivos e diretórios.

Embora seja possível fazer login sem ter um diretório *home*, isso limitará as variáveis de ambiente pessoal disponíveis para o usuário, o que significa dizer que configurações personalizadas de shell, preferências de aplicativos e outras personalizações podem não estar disponíveis se o diretório *home* não estiver configurado corretamente.

Em resumo, adotar senhas fortes e garantir que cada usuário tenha um diretório *home* configurado adequadamente são práticas importantes para garantir a segurança e a personalização do sistema.

6.2 Os arquivos de configuração

Há quatro arquivos básicos que dizem respeito à administração de usuários:

Arquivo	Descrição
/etc/passwd	Contém as informações dos usuários.

Arquivo	Descrição
/etc/shadow	Contém as informações das senhas dos usuários.
/etc/group	Contém as informações dos grupos e usuários que fazem parte deles.
/etc/gshadow	Contém informações a respeito das senhas de grupos.

Cada usuário no sistema é identificado por uma linha no arquivo `/etc/passwd`, sendo os campos separados pelo caractere dois pontos (:). Por exemplo:

```
jane:x:1001:1001:Jane Smith:/home/jane:/bin/bash
```

As informações contidas nessa linha são:

- Login do usuário;
- Referência da senha (armazenada em `/etc/shadow`);
- UID (*User identify*) — Número de identificação do usuário;
- GID (*Group identify*) — Identificação do grupo primário;
- Comentários e informações adicionais sobre o usuário;
- Diretório pessoal;
- Shell do usuário.

O arquivo `/etc/shadow` é o local onde as senhas são armazenadas de forma criptografada. Ele possui permissões de leitura restritas apenas ao usuário *root*, garantindo maior segurança.

No entanto, em servidores mais antigos, pode ser que as senhas estejam armazenadas diretamente no arquivo `/etc/passwd`. Para migrar as senhas para o arquivo `/etc/shadow` e adotar uma abordagem mais segura, você pode utilizar o seguinte comando:

```
# pwconv
```

Esse comando moverá as senhas do arquivo `/etc/passwd` para o arquivo `/etc/shadow`, garantindo uma camada adicional de proteção.

É importante ressaltar que é recomendado armazenar as senhas no arquivo `/etc/shadow`, pois o armazenamento direto no arquivo `/etc/passwd` pode ser potencialmente inseguro.

Caso queira fazer esta ação (não recomendada e insegura) por algum motivo, existe o comando:

```
# pwunconv
```

6.3 Adicionar contas de usuários

No Linux, podemos adicionar novos usuários através do comando `useradd`, mas este comando só é possível caso seu usuário tenha privilégio administrativo, como o *root*. A sintaxe básica do comando é:

```
# useradd juliano
```

Após adicionar um usuário, você deve definir a sua senha. Essa tarefa é realizada com o comando `passwd`:

```
# passwd juliano
```

Para criar um usuário com o comando `useradd` e definir uma senha ao mesmo tempo, você pode usar a opção `-p` para especificar a senha criptografada. Aqui está um exemplo de como criar um usuário com o comando `useradd` e definir uma senha:

```
# useradd -m -p <senha_criptografada> <nome_do_usuário>
```

Certifique-se de substituir `<senha_criptografada>` pela senha desejada, criptografada usando um algoritmo como o *crypt* ou o *openssl passwd*. Além disso, substitua `<nome_do_usuário>` pelo nome do usuário que você deseja criar.

Aqui está um exemplo de como criar um usuário chamado "*john*" com a senha "*mypass*":

```
# useradd -m -p $(openssl passwd -1 mypass) john
```

O uso de `$(openssl passwd -1 mypass)` gera a senha criptografada correspondente a "*mypass*" usando o algoritmo de criptografia *openssl*. Lembre-se de executar o comando como superusuário para ter permissões suficientes para criar um novo usuário.

6.4 Modificar contas de usuários

Às vezes, precisamos alterar um atributo de uma conta de usuário existente, como o nome de login, o shell de login, a data de expiração da senha, entre outros. Nesses casos, empregamos o comando `usermod`. Vejamos alguns exemplos de uso do comando `usermod`. Podemos utilizá-lo para alterar o diretório *home* de um usuário:

```
sudo usermod -d /caminho/novo/diretorio username
```

Isso alterará o diretório *home* do usuário especificado pelo nome de usuário `username` para `/caminho/novo/diretorio`.

Podemos usar o `usermod` para alterar o shell padrão de um usuário:

```
sudo usermod -s /bin/zsh username
```

Isso modificará o shell padrão do usuário `username` para `/bin/zsh`.

Para adicionar um usuário a um grupo adicional usando o `usermod`, fazemos:

```
sudo usermod -aG grupo username
```

Isso adicionará o usuário `username` ao grupo `grupo` sem alterar o grupo primário do usuário.

Para alterar o nome de usuário com `usermod`:

```
sudo usermod -l novo-username username
```

Isso renomeará o usuário `username` para `novo-username`.

Podemos usar o `usermod` para definir uma data de expiração para a conta de um usuário:

```
sudo usermod -e YYYY-MM-DD username
```

Isso definirá uma data de expiração para a conta do usuário `username`; após a data definida, o usuário não poderá fazer login com aquele nome de usuário.

Para bloquear uma conta de usuário:

```
sudo usermod -L username
```

Isso bloqueará a conta do usuário `username`, impedindo-o de fazer login.

Esses são apenas alguns exemplos de uso do comando `usermod` para modificar os atributos de um usuário existente no sistema Linux. Lembre-se de executar o comando com privilégios de superusuário (usando `sudo`) para realizar as alterações necessárias.

6.5 Excluir contas de usuário

O comando para excluir uma conta de usuário no Linux é o `userdel`. Com a opção `-r`, ele também remove o diretório pessoal do usuário e todo o seu conteúdo. Outros arquivos, localizados em outros diretórios, devem ser buscados e excluídos manualmente.

```
userdel -r juliano
```

Para buscar arquivos de um usuário específico em todo o sistema, você pode usar o seguinte comando:

```
find / -user <nome_do_usuario>
```

Certifique-se de substituir `<nome_do_usuário>` pelo nome de usuário do qual você deseja buscar os arquivos.

6.6 Adicionar, modificar e excluir grupos

Criar grupos no Linux é uma prática comum e útil por algumas razões:

- **Gerenciamento de permissões:** os grupos permitem que você gerencie as permissões de acesso aos arquivos e diretórios de forma mais eficiente. Em vez de atribuir permissões individualmente para cada usuário, você pode adicionar usuários a grupos específicos e, em seguida, definir permissões para o grupo. Isso facilita a atribuição de permissões consistentes para um conjunto de usuários.
- **Colaboração:** grupos no Linux são frequentemente usados para facilitar a colaboração entre usuários. Por exemplo, se várias pessoas estão trabalhando em um projeto, você pode criar um grupo específico para esse projeto e adicionar os membros relevantes a ele. Dessa forma, eles terão acesso aos mesmos arquivos e diretórios, facilitando o compartilhamento e a edição colaborativa.
- **Administração do sistema:** os grupos também são úteis para fins de administração do sistema. Em vez de gerenciar cada usuário individualmente, você pode criar grupos com diferentes níveis de acesso e atribuir usuários a esses grupos. Por exemplo, você pode ter um grupo de administradores do sistema, com privilégios mais elevados, e um grupo de usuários

comuns, com privilégios restritos. Isso facilita a administração do sistema e a aplicação de políticas de segurança.
- **Segurança:** o uso de grupos pode melhorar a segurança do sistema. Ao atribuir permissões de arquivo e diretório com base em grupos, você pode garantir que apenas os usuários relevantes tenham acesso a determinados recursos. Isso reduz a chance de erros de permissão acidentais e ajuda a proteger os dados do sistema contra acesso não autorizado.

O comando básico para se criar um grupo no Linux é o `groupadd`, e sua sintaxe básica é:

```
# groupadd [opcoes] nome_do_grupo
```

As opções mais comuns são:

- `-g GID` ou `--gid GID`: especifica o ID do grupo (GID) manualmente. Por padrão, o próximo ID de grupo disponível é atribuído automaticamente;
- `-r` ou `--system`: cria um grupo de sistema. Grupos de sistema têm IDs de grupo menores que 1000 e são normalmente usados para fins específicos do sistema, em vez de usuários normais;
- `-f` ou `--force`: força a criação do grupo, mesmo se o nome já estiver sendo usado por outro grupo;
- `-K chave=valor` ou `--key chave=valor`: define uma opção de configuração específica para o grupo. Por exemplo, `-K GID_MIN=1000` define o valor mínimo do ID de grupo permitido.

No exemplo a seguir, é criado o grupo `desenvolvimento` com o GID `1090`:

```
# groupadd -g 1090 desenvolvimento
```

6.7 Renomear o grupo

Caso exista a necessidade de renomear o grupo, pode-se utilizar o seguinte comando:

```
# groupmod -n desenvolvimento-web -g 1080 desenvolvimento
```

Nesse código, o grupo `desenvolvimento`, que criamos no exemplo anterior, tem seu nome alterado para `desenvolvimento-web` e o seu GID também alterado de `1090` para `1080`.

6.8 Adicionar usuários a um grupo

Agora que já possuímos um grupo, podemos adicionar usuários a ele. Isso pode ser feito com o seguinte comando:

```
usermod -aG grupo <nome_do_usuario>
```

Nesse código:

- `-a` indica que você deseja adicionar o usuário ao grupo (em vez de substituir os grupos existentes);
- `-G` especifica o grupo ao qual você deseja adicionar o usuário;
- `grupo` é o nome do grupo ao qual você deseja adicionar o usuário;
- `nome_do_usuario` é o nome do usuário que você deseja adicionar ao grupo.

Por exemplo, para adicionar o usuário " `juliano` " ao grupo " `desenvolvimento` ", podemos usar o seguinte comando:

```
usermod -aG desenvolvimento juliano
```

O arquivo que armazena as informações dos grupos é o `/etc/group`. Ele tem as linhas parecidas com o `/etc/passwd`:

```
developers:x:1001:john,jane
```

Nesse exemplo, `developers` é o nome do grupo, `x` indica que há uma senha definida, `1001` é o GID atribuído ao grupo e `john` e `jane` são os usuários que pertencem a esse grupo.

Para remover um grupo, usamos o comando `groupdel`. Por exemplo:

```
# groupdel desenvolvimento-web
```

6.9 O diretório esqueleto

O diretório `/etc/skel` (*skel* de *skeleton*, esqueleto em inglês) é um diretório modelo que contém arquivos e diretórios que são copiados para o diretório *home* de um novo usuário criado no sistema. Por exemplo, se você criar no diretório `/etc/skel/` um diretório chamado `Administracao` e adicionar um novo usuário ao sistema de nome `oliveira`, este diretório será copiado para `/home/oliveira/Administracao`.

Neste diretório esqueleto, você também pode personalizar alguns arquivos de configuração que serão replicados aos novos usuários. São eles:

- `.bashrc`: este é um arquivo de configuração do shell Bash, que é executado quando um usuário faz login no sistema. Ele pode conter configurações personalizadas para o ambiente de shell do usuário, como variáveis de ambiente, *aliases* e funções;
- `.bash_profile`: este é outro arquivo de configuração do shell Bash, que é executado após o `.bashrc`. Ele também pode conter configurações personalizadas, mas é executado apenas uma vez no login do usuário, enquanto o `.bashrc` é executado em cada nova sessão do shell;
- `.profile`: este é um arquivo de configuração geral do shell que é lido por vários tipos de shell no Linux. Ele também pode

conter configurações personalizadas para o ambiente do usuário;
- `.bash_logout` : este arquivo é executado quando o usuário faz logout do sistema. Ele pode ser usado para executar tarefas específicas de limpeza ou ações finais antes que o usuário saia.

Vale lembrar que os arquivos e diretórios do `/etc/skel` são apenas modelos e podem ser modificados por um administrador de sistema para atender às necessidades da configuração do sistema. Os arquivos contidos no diretório `/etc/skel` são copiados apenas quando um novo usuário é criado e não afetam usuários existentes no sistema.

Conclusão

A administração de usuários no Linux é uma tarefa fundamental para as pessoas administradoras de sistemas. No contexto do exame da LPIC-1, esse tópico é de grande importância, uma vez que pode conter até cinco questões valiosas na prova.

Através do gerenciamento eficaz das contas de usuário, é possível promover a produtividade, garantir a segurança dos dados e manter a integridade do sistema. Ao criar e configurar contas de usuário, é possível atribuir permissões adequadas, controlar o acesso aos recursos e estabelecer políticas de segurança, como exigência de senhas fortes e expiração de senhas.

Além disso, a administração de usuários permite agrupá-los de acordo com departamentos ou equipes de trabalho, simplificando o gerenciamento de permissões e o compartilhamento de recursos entre os membros do grupo.

Ao administrar os usuários de forma eficiente, administradores podem desativar ou remover contas de usuário não utilizadas, fazer ajustes nas configurações de conta, restringir privilégios de

superusuário (*root*) e garantir a conformidade com políticas de segurança e regulamentações.

Em resumo, a administração de usuários no Linux é um aspecto essencial para administradores de sistemas e é um tópico de grande relevância no exame da LPIC-1. Gerenciar as contas de usuário de forma adequada proporciona aumento da produtividade, proteção dos dados e preservação da integridade do sistema operacional Linux.

CAPÍTULO 7
Tópico 107.2 – Automatizar e agendar tarefas administrativas de sistema

Certificação	LPIC-1
Versão	5.0
Tópico no exame	107.2 – Automatizar e agendar tarefas administrativas de sistema
Objetivo	Os candidatos devem ser capazes de usar `cron` e `timers` do `systemd` para executar tarefas em intervalos regulares e usar `at` para rodar tarefas em um horário específico.
Peso	4

Um bom administrador ou uma boa administradora de sistemas deve ter conhecimento sobre o agendador de tarefas do Linux, conhecido como **cron**. Esse aplicativo é essencial para evitar a execução manual de tarefas repetitivas. Com o cron, é possível organizar e agendar tarefas futuras de forma automatizada, definindo quando e com que frequência elas devem ser executadas.

O cron é uma ferramenta extremamente flexível e poderosa, pois permite configurar tarefas para serem executadas em intervalos regulares: diariamente, semanalmente, mensalmente ou em dias específicos da semana. Além disso, é possível especificar o horário exato em que uma tarefa deve ser executada.

Uma característica importante do cron é a capacidade de executar comandos ou scripts, o que torna possível a automação de diversas tarefas administrativas, como fazer backup dos arquivos do sistema regularmente, enviar relatórios por e-mail ou executar rotinas de manutenção programadas. O cron também permite o uso de

variáveis de ambiente, o que facilita a configuração de tarefas que exigem um contexto específico. Essas variáveis fornecem informações como o diretório atual, o usuário que executou o cron e outros detalhes relevantes.

Em resumo, o conhecimento sobre o agendador de tarefas do Linux, o cron, é fundamental para ser um(a) bom(a) administrador(a) de sistemas. Com essa aplicação, é possível evitar a realização manual de tarefas repetitivas, melhorar a eficiência e automatizar processos, contribuindo para um ambiente de administração de sistemas mais organizado e produtivo.

7.1 Como agendar jobs com o cron

O cron é um *daemon* que roda continuamente e desperta a cada um minuto para verificar uma série de tabelas em busca de tarefas a executar. Essas tabelas são chamadas de *crontabs* e contêm os chamados *cron jobs* (trabalhos do cron). Os usuários do Linux podem definir suas próprias *crontabs*. Para isso, abra um terminal de comando e digite:

```
$ crontab -e
```

Esse comando abre o arquivo para edição dos *cron jobs*. Se for a primeira vez que você o está utilizando, será solicitada a seleção do editor de texto padrão. Escolha o editor com o qual você tem mais familiaridade (por exemplo, *Nano* ou *Vim*) e prossiga.

A sintaxe básica para uma linha *crontab* é:

```
 * * * * * comando
```

Os cinco campos separados por espaços representam, respectivamente:

- Minuto (0-59): o minuto exato em que o trabalho deve ser executado;
- Hora (0-23): a hora exata em que o trabalho deve ser executado;
- Dia do mês (1-31): a dia específico do mês em que o trabalho deve ser executado;
- Mês (1-12): o mês específico em que o trabalho deve ser executado;
- Dia da semana (0-7): a dia específico da semana em que o trabalho deve ser executado (0 e 7 representam o domingo).

Além disso, existem alguns caracteres especiais que podem ser usados para representar intervalos ou listas de valores:

- Asterisco (`*`): significa "todos os valores possíveis" para o campo correspondente;
- Hífen (`-`): indica um intervalo de valores. Por exemplo, `1-5` representa os valores de 1 a 5;
- Vírgula (`,`): é usada para separar valores individuais. Por exemplo, `1,3,5` representa os valores 1, 3 e 5;
- Barra (`/`): indica um intervalo incremental. Por exemplo, `*/10` no campo de minutos significa "a cada 10 minutos".

Alguns exemplos:

Agendamento:	Comando:
Executar um script todos os dias às 8h da manhã	`0 8 * * * /caminho/para/meu_script.sh`
Executar um backup toda segunda-feira às 2h da manhã	`0 2 * * 1 /caminho/para/backup.sh`
Executar um comando a cada hora	`0 * * * * comando`
Executar um script todas as noites às 23h30	`30 23 * * * /caminho/para/meu_script.sh`

Agendamento:	Comando:
Executar um comando a cada 15 minutos	`*/15 * * * * comando`
Executar um script no dia 1º de cada mês às 12h	`0 12 1 * * /caminho/para/meu_script.sh`
Executar um comando a cada minuto	`* * * * * comando`

Você pode visualizar os agendamentos com o seguinte comando:

```
crontab -l
```

Os agendamentos ficam localizados em `/var/spool/cron/crontabs`. Para remover todos os agendamentos do usuário, execute:

```
crontab -r
```

7.2 Agendamento de uma única tarefa

Para agendarmos uma única tarefa, o comando utilizado é o `at`. Sua sintaxe básica é:

```
# at HH:mm MM/DD/YYYY
at> tar zcvf /backup/backup-etc.tar.gz /etc/
at> (crtl + d)
```

Onde `HH:mm` representa a hora e o minuto e `MM/DD/YYYY` representa o mês/dia/ano. Após digitar o comando que será executado, finalize com `ctrl+d`.

Você pode verificar as tarefas agendadas com o seguinte comando:

```
# atq
```

O diretório que guarda os agendamentos do `at` é:

```
# cd /var/spool/cron/atjobs
```

Para remover um agendamento, utilize o seguinte comando:

```
# atrm [numero-do-agendamento]
```

Conclusão

O agendamento de tarefas por meio do *crontab* desempenha um papel fundamental na automação e eficiência das operações em sistemas Linux. A capacidade de agendar a execução de comandos ou scripts em momentos específicos proporciona uma série de benefícios e simplifica a administração do sistema. Aqui estão algumas conclusões sobre a importância do agendamento de tarefas com o *crontab* no Linux:

- Automação: o *crontab* permite automatizar tarefas rotineiras, como backups, limpezas de diretórios, atualizações de banco de dados, entre outros. Com o agendamento, essas tarefas podem ser executadas regularmente sem intervenção manual, economizando tempo e esforço da pessoa administradora do sistema;
- Conveniência: o agendamento de tarefas oferece a conveniência de realizar operações em horários específicos, mesmo quando o usuário não está presente ou está ocupado com outras atividades. Isso garante que as tarefas sejam executadas pontualmente, independentemente da disponibilidade do usuário;
- Redução de erros: ao utilizar o *crontab* para agendar tarefas, evita-se a necessidade de executar manualmente comandos ou scripts em momentos específicos. Isso reduz o risco de erros humanos, como esquecer de executar uma tarefa importante ou executá-la em um momento inadequado;

- Otimização do desempenho: com o agendamento de tarefas, é possível executar processos de manutenção, otimização e sincronização durante períodos de baixa atividade do sistema. Dessa forma, os recursos do sistema são utilizados de maneira mais eficiente, evitando sobrecargas desnecessárias durante os horários de pico de uso;

- Monitoramento e alertas: o *crontab* pode ser utilizado para executar scripts de monitoramento que verificam a saúde do sistema, o uso de recursos ou outras métricas relevantes. Isso permite a detecção de problemas em tempo real e o envio de alertas para que medidas corretivas sejam tomadas.

- Planejamento e escalonamento: ao agendar tarefas com o *crontab*, é possível planejar e escalonar a execução de processos e rotinas de acordo com as necessidades do sistema e dos usuários. Isso ajuda a distribuir a carga de trabalho de forma equilibrada e evitar gargalos ou congestionamentos.

Em resumo, o agendamento de tarefas com o *crontab* no Linux oferece uma maneira poderosa de automatizar, otimizar e simplificar as operações do sistema. Ao programar tarefas recorrentes, o *crontab* contribui para a eficiência, confiabilidade e estabilidade do sistema, permitindo que administradores se concentrem em outras atividades mais estratégicas.

CAPÍTULO 8
Tópico 107.3 – Localização e internacionalização

Certificação	LPIC-1
Versão	5.0
Tópico no exame	107.3 – Localização e internacionalização
Objetivo	Os candidatos devem ser capazes de localizar um sistema em um idioma diferente do inglês, bem como entender por que `LANG=C` é útil quando estiverem escrevendo scripts.
Peso	3

O Linux é um sistema acessível e amigável para usuários de diferentes partes do mundo, pois ele permite a *localização* e *internacionalização*.

A **localização** refere-se à personalização do sistema operacional para se adequar às características de uma determinada localidade, como idioma, formato de data e hora, moeda, sistema de numeração, entre outras. Isso envolve a tradução de mensagens e textos exibidos pelo sistema, a adaptação de formatos de data e hora, a utilização de caracteres específicos de um idioma e a configuração de preferências regionais. A localização permite que o sistema seja utilizado de forma mais natural e familiar por usuários de diferentes regiões.

A **internacionalização**, por sua vez, é o processo de projetar e desenvolver um sistema operacional de forma que ele possa ser facilmente adaptado para diferentes localizações. Isso envolve a separação de recursos localizáveis do código-fonte, a utilização de

padrões e bibliotecas que suportam caracteres e conjuntos de caracteres internacionais e a criação de interfaces flexíveis que possam acomodar diferentes idiomas e culturas. A internacionalização permite que um sistema seja facilmente localizado para diferentes idiomas e regiões sem a necessidade de alterações significativas em seu código-fonte.

Em sistemas Linux, a localização e internacionalização são suportadas por meio de um conjunto de ferramentas e bibliotecas conhecido como **GNU** `gettext`. O `gettext` fornece uma maneira padronizada de criar e gerenciar arquivos de tradução que contêm as mensagens do sistema em diferentes idiomas. Além disso, existem variáveis de ambiente, como `LANG` e `LC_ALL`, que podem ser configuradas para definir o idioma e a localização padrão do sistema.

8.1 Configurando o fuso horário

O Linux oferece diversas ferramentas de suporte a idiomas, do inglês a lugares fora do eixo EUA/Europa. Estas configurações alteram idioma, fuso horário, moeda e definição do teclado. O comando `tzselect` exibirá os fusos horários (*time zones*) oficiais e, após a escolha, criará o arquivo `/etc/timezone` contendo as informações de fuso horário. Ao digitar o comando `tzselect`, serão apresentadas as seguintes opções:

```
Please identify a location so that time zone rules can be set correctly.
Please select a continent or ocean.
 1) Africa
 2) Americas
 3) Antarctica
 4) Arctic Ocean
 5) Asia
 6) Atlantic Ocean
 7) Australia
 8) Europe
```

```
 9) Indian Ocean
10) Pacific Ocean
11) none - I want to specify the time zone using the Posix TZ format.
# Digite o número correspondente à sua região geográfica e pressione
Enter.
```

Nessa etapa, vamos escolher a opção 2, *Americas*. Após a seleção, serão exibidas as seguintes opções de países:

```
Please select a country.
 1) Argentina
 2) Brazil
 3) Canada
 4) Chile
 5) United States
 6) Mexico
# Digite o número correspondente ao país e pressione Enter.
```

Suponha que selecionamos a opção 2, *Brazil*. Na próxima etapa, você deve especificar a região. Escolha a região *Eastern* (leste). Após isto, ele mostra um resumo:

```
The following information has been given:
  Brazil
  Eastern (ET)

Therefore TZ='America/Sao_Paulo' will be used.
Local time is now: Mon Jul 5 14:30:00 BRT 2023.
Universal Time is now: Mon Jul 5 17:30:00 UTC 2023.
Is the above information OK?
1) Yes
2) No
```

Se estiver tudo correto, confirme com a opção 1. O comando `tzselect` vai gerar o arquivo `/etc/timezone` com as configurações definidas:

```
# cat /etc/timezone
America/Sao_paulo
```

Todos os arquivos de horário de verão disponíveis no sistema ficam em:

```
# cd /usr/share/zoneinfo/
```

8.2 A variável LANG

O idioma do sistema pode ser alterado através da variável `LANG`. Para visualizar o valor atual da variável `LANG`, abra o terminal e digite o seguinte comando:

```
$ echo $LANG
```

O valor retornado será o idioma atualmente definido. Por exemplo, `en_US.UTF-8` para inglês nos Estados Unidos.

Para alterar o valor da variável `LANG`, você pode usar o comando `export` seguido pelo novo valor desejado. Por exemplo, se você quiser definir o idioma para português do Brasil:

```
$ export LANG=pt_BR.UTF-8
```

Isso define a variável `LANG` como `pt_BR.UTF-8`, que é o código para português do Brasil. Você pode verificar se a variável `LANG` foi alterada executando novamente o comando:

```
$ echo $LANG
```

O valor retornado agora será o novo idioma definido. Observe que essa alteração afetará apenas a sessão atual do terminal em que o comando foi executado.

Para tornar a alteração permanente, você pode adicionar o comando `export LANG=pt_BR.UTF-8` ao seu arquivo de perfil de usuário, como `/.bashrc` ou `/.profile`. Isso garantirá que a variável `LANG` seja configurada corretamente sempre que você fizer login no sistema.

Além da variável `LANG`, outras variáveis de ambiente afetam aspectos específicos da localidade, como o símbolo monetário ou o separador de milhar correto para números:

- `LC_CTYPE`: define como o sistema tratará certos conjuntos de caracteres. Ele define, por exemplo, quais caracteres considerar como maiúsculas ou minúsculas;
- `LC_MESSAGES`: define o idioma para exibir as mensagens de programas (principalmente programas do GNU);
- `LC_MONETARY`: define a unidade monetária e o formato da moeda;
- `LC_NUMERIC`: define o formato numérico para valores não monetários. Sua finalidade principal é definir os separadores de milhar e decimais;
- `LC_TIME`: define o formato de hora e data;
- `LC_PAPER`: define o tamanho padrão do papel;
- `LC_ALL`: sobrepõe todas as outras variáveis, incluindo `LANG`.

O comando `locale` mostra todas as variáveis definidas na configuração de localidade atual:

```
$ locale
LANG=pt_BR.UTF-8
LC_CTYPE="pt_BR.UTF-8"
LC_NUMERIC=pt_BR.UTF-8
LC_TIME=pt_BR.UTF-8
LC_COLLATE="pt_BR.UTF-8"
LC_MONETARY=pt_BR.UTF-8
LC_MESSAGES="pt_BR.UTF-8"
LC_PAPER=pt_BR.UTF-8
LC_NAME=pt_BR.UTF-8
LC_ADDRESS=pt_BR.UTF-8
LC_TELEPHONE=pt_BR.UTF-8
LC_MEASUREMENT=pt_BR.UTF-8
LC_IDENTIFICATION=pt_BR.UTF-8
LC_ALL=
```

Lembre-se de substituir `pt_BR.UTF-8` pelo código do idioma desejado, conforme a convenção de códigos de idioma do seu sistema

operacional Linux.

8.3 Conversão de codificação

Quando você visualiza um texto que foi criado em um sistema com uma codificação diferente, os caracteres podem parecer ilegíveis. O comando `iconv` ajuda a resolver esse problema, convertendo o arquivo de sua codificação de caracteres original para a desejada. Por exemplo, para converter um arquivo `documento.txt` da codificação ISO-8859-1 para `documento2.txt` utilizando UTF-8, use o comando:

```
$ iconv -f ISO-8859-1 -t UTF-8 documento.txt > documento2.txt
```

Para listar todas as codificações aceitas pelo comando `iconv`, execute o comando `$ iconv -l`, ou `$ iconv --list`.

8.4 Comando date

O comando `date` exibe o fuso horário atualmente configurado. Por exemplo:

```
$ date
Mon Oct 12 10:20:10 -03 2022
```

Na saída desse exemplo, o `-03` indica que a hora exibida é três horas a menos que a UTC (Tempo Universal Coordenado). UTC é uma referência de tempo internacionalmente aceita e utilizada como referência para o controle do tempo em todo o mundo.

8.5 Comando timedatectl

O comando `timedatectl` é disponível para distribuições que usam o `systemd`. Ele mostra detalhes das configurações de data e hora:

```
$ timedatectl
Local time: Sat 2019-10-19 17:53:18 -03
  Universal time: Sat 2019-10-19 20:53:18 UTC
  RTC time: Sat 2019-10-19 20:53:18
  Time zone: America/Sao_Paulo (-03, -0300)
  System clock synchronized: yes
systemd-timesyncd.service active: yes
  RTC in local TZ: no
```

Conforme observado na seção de Zona Horária, também são aceitos nomes de fusos horários com base em localizações específicas, como `America/Sao_Paulo`. O fuso horário padrão para o sistema é gerenciado no arquivo `/etc/timezone` e pode ser especificado usando tanto o nome descritivo quanto o deslocamento em relação ao UTC.

Os nomes genéricos de fusos horários que indicam a diferença em relação ao UTC devem começar com `Etc`. Portanto, para estabelecer o fuso horário padrão como GMT+3, por exemplo, é necessário usar o nome de fuso horário `Etc/GMT+3`.

```
$ cat /etc/timezone
Etc/GMT+3
```

Embora os nomes de fusos horários com base em localidades não exijam a indicação da diferença de hora para funcionar, eles não são tão fáceis de escolher. A mesma zona pode ter mais de um nome, o que pode tornar mais difícil de lembrar. Para mitigar esse problema, o comando `tzselect`, que abordamos neste capítulo, é a melhor opção para configurar o fuso horário, já que é interativo.

Conclusão

Este é um tópico relativamente fácil para o exame LPIC-1 102-500 e, como é de peso três, é uma boa oportunidade para ganhar alguns

pontos. Leia atentamente a questão dos nomes das variáveis e teste algumas definições de fuso horário para fixar o conhecimento, dando atenção especial a estes arquivos e comandos:

- `/etc/timezone`
- `/etc/localtime`
- `/usr/share/zoneinfo/`
- `locale`
- `tzselect`
- `iconv`
- `date`

Tópico 108: Serviços essenciais do sistema

Capítulo 9
Tópico 108.1 – Manutenção da data e hora do sistema

Certificação	LPIC-1
Versão	5.0
Tópico no exame	108.1 – Manutenção da data e hora do sistema
Objetivo	Os candidatos devem ser capazes de manter correta a hora do sistema e de sincronizar o relógio através de NTP.
Peso	3

Uma tarefa crucial para o bom funcionamento de um sistema computacional é manter o controle do tempo de forma precisa. A prática de controlar a hora parece trivial para o usuário final, mas o sistema precisa ser capaz de lidar com muitas idiossincrasias e casos extremos de forma inteligente. Os fusos horários não são estáticos, eles são alterados por uma decisão administrativa ou política — no Brasil, por exemplo, já tivemos diversas mudanças em relação ao horário de verão. O Linux possui ferramentas robustas para ajustar o horário e estas variáveis que podem ocorrer.

Quando o Linux é iniciado, o *relógio do sistema* começa a marcar o tempo, e ele é controlado pelo sistema operacional. Além disso, os computadores possuem o *relógio de hardware* ou de *tempo real* (RTC). Esse relógio é um recurso da placa-mãe que marca o tempo

do seu computador, esteja ele ligado ou não. A maioria das distribuições de Linux permite que você configure o seu sistema de data e hora pela rede, por meio de um protocolo chamado de NTP (*Network Time Protocol*). Na maioria dos casos, a única configuração que um usuário precisa fazer é definir o seu fuso horário e o NTP configura todo o resto.

9.1 Horário local e horário universal

O relógio do sistema é configurado para o Tempo Universal Coordenado (UTC), que é a hora local de Greenwich, no Reino Unido. Para converter UTC em horário local, é necessário adicionar ou subtrair o fuso horário e o horário de verão, se aplicável. No Linux, o utilitário nativo do sistema para visualizar o horário local é o comando `date` :

```
$ date
Tue Sep 12 10:47:05 AM UTC 2023
```

O comando `date` possui algumas opções, como exibir o horário UTC atual:

```
$ date -u
```

No exame de certificação não é necessário que você conheça todas as opções do comando `date` , mas é sempre bom ler a página de ajuda, através do comando:

```
$ date --help
```

9.2 Relógio de hardware

Para visualizar a hora do relógio mantido em tempo real, usamos o comando `hwclock`. Esse comando exige privilégios elevados:

```
# hwclock
```

O comando `hwclock` tem uma variedade de opções que permitem visualizar, configurar e sincronizar o RTC. As opções mais comuns incluem:

- `-r` ou `--show`: exibe a hora atual do RTC;
- `-s` ou `--set`: define a hora do RTC para o valor especificado;
- `-w` ou `--systohc`: sincroniza o RTC com o relógio do sistema;
- `-c` ou `--adjust`: ajusta o RTC para compensar o desvio do relógio do sistema.

Aqui estão alguns exemplos de como usar o comando `hwclock`:

```
# Para exibir a hora atual do RTC:
hwclock -r

# Para definir a hora do RTC para 2023-09-12 12:00:00:
hwclock -s 2023-09-12 12:00:00

# Para sincronizar o RTC com o relógio do sistema:
hwclock -w

# Para ajustar o RTC para compensar o desvio do relógio do sistema:
hwclock -c
```

9.3 O comando timedatectl

O comando `timedatectl` permite que você visualize e configure o relógio do sistema, o fuso horário e o horário de verão. Entre as suas opções:

- `status` : exibe o estado atual do relógio do sistema, incluindo a hora, o fuso horário e o horário de verão;
- `set-time` : define a hora do sistema para o valor especificado;
- `set-timezone` : define o fuso horário do sistema para o valor especificado;
- `set-local-time` : sincroniza o relógio do sistema com a hora local;
- `set-utc` : sincroniza o relógio do sistema com o UTC.

Alguns exemplos de uso:

```
# Para exibir o status atual do relógio do sistema:
timedatectl status

# Para definir a hora do sistema para 2023-09-12 12:00:00:
timedatectl set-time 2023-09-12 12:00:00

# Para definir o fuso horário do sistema para América/São Paulo:
timedatectl set-timezone America/Sao_Paulo

# Para sincronizar o relógio do sistema com a hora local:
timedatectl set-local-time

# Para sincronizar o relógio do sistema com o UTC:
timedatectl set-utc
```

9.4 O servidor NTP

Quando possuímos computadores em rede, é necessário sincronizar os horários; para isso, usamos o servidor NTP. O arquivo `/etc/ntp.conf` contém informações de configuração sobre como o sistema realiza sua sincronização ao tempo de rede. Por padrão, os servidores NTP são especificados em uma seção como esta:

```
# Use public servers from the pool.ntp.org project.
# Please consider joining the pool (http://www.pool.ntp.org/join.html).
server 0.centos.pool.ntp.org iburst
server 1.centos.pool.ntp.org iburst
server 2.centos.pool.ntp.org iburst
server 3.centos.pool.ntp.org iburst
```

A sintaxe para adicionar servidores NTP é realizada desta forma:

```
server (Endereço IP)
server server.url.localhost
```

Você pode configurar com uma URL (desde que seu DNS esteja correto) ou via IP:

```
server 192.168.0.1 iburst
```

Usando servidor da web ou local?

Os servidores NTP usados por padrão são um projeto de código aberto.

Em muitos casos, "atualizar sua hora através dos servidores on-line do NTP (pool.ntp.org) pode ser a solução", porém vale lembrar que, para que eles funcionem, é necessário ter uma conexão com a internet e confiar na disponibilidade dos servidores `ntp.org`. Se você necessita de algo mais preciso, configurar um servidor NTP local pode ser uma solução mais eficaz.

Uma forma de sincronizar seu horário com os servidores `ntp.org` é usando o comando `ntpdate`. Por exemplo:

```
ntpdate 0.pool.ntp.org
```

Nesse caso, o relógio do sistema será sincronizado com o servidor NTP `0.pool.ntp.org`.

ntpq

Para monitorar o status do NTP, usamos o comando `ntpq`:

```
# ntpq -p

remote        refid        st  t  when poll reach delay offset  jitter
*192.168.1.1  192.168.1.1  2   u  188  64   377   0.043 0.000   0.017
```

O comando `ntpq -p` exibe informações sobre o estado atual do *daemon* NTP, incluindo o *stratum*, o *offset* e o *delay*. O ***stratum*** é um número que indica o nível de sincronização do relógio do sistema com o servidor NTP. O ***offset*** é a diferença entre o tempo do sistema e o tempo do servidor NTP. O ***delay*** é o tempo que leva para um pacote NTP viajar do sistema para o servidor NTP, e vice-versa.

Nesse exemplo, o relógio do sistema está sincronizado com o servidor NTP 192.168.1.1. O *stratum* (*st*) do servidor NTP é 2, o que significa que está sincronizado com um servidor NTP de nível 1 ou 0. O delay é de 0.043 segundos, o que significa que o relógio do sistema está 43 milissegundos atrasado em relação ao servidor NTP. O Jitter é de 0.017 segundos, o que significa que leva 17 milissegundos para um pacote NTP viajar do sistema para o servidor NTP e vice-versa.

Conclusão

É importante manter a hora do sistema Linux sincronizada por várias razões, incluindo:

- Garantir a precisão do tempo: quando o relógio do sistema está sincronizado, você pode ter certeza de que as informações de tempo são precisas. Isso é importante para uma variedade de tarefas, tais como sincronizar arquivos, agendar tarefas e acessar serviços on-line.
- Evitar problemas de sincronização: quando o relógio do sistema não está sincronizado, podem ocorrer problemas de sincronização com outros dispositivos e sistemas. Isso pode levar a erros, perda de dados e outros problemas.

- Melhorar a segurança: alguns serviços on-line e aplicativos dependem de uma hora precisa para funcionar corretamente. Quando o relógio do sistema não está sincronizado, esses serviços e aplicativos podem não funcionar corretamente. Isso pode tornar seu sistema mais vulnerável a ataques.

No exame LPIC-1 você terá pelo menos três questões sobre como configurar a data e hora do sistema. Sendo assim, considere revisar e treinar o conteúdo deste capítulo.

Capítulo 10
Tópico 108.2 – Log do sistema

Certificação	LPIC-1
Versão	5.0
Tópico no exame	108.2 – Log do sistema
Objetivo	Os candidatos devem ser capazes de configurar o serviço *rsyslog*. Este objetivo inclui configurar o serviço de mensagens de log para enviar as mensagens para um servidor central ou receber as mensagens como um servidor central. O uso do subsistema de *journal* do `systemd` é cobrado.
Peso	4

Os **logs** são arquivos de texto que registram eventos do sistema, como erros, avisos e informações. Eles são essenciais para administradores de sistema, pois ajudam a diagnosticar problemas, solucionar incidentes e monitorar o desempenho do sistema. Os logs podem registrar uma ampla gama de informações, incluindo:

- Tentativas de autenticação malsucedidas;
- Erros em programas e serviços;
- Hosts bloqueados pelo firewall;
- Processos corrompidos.

Uma boa administradora de sistemas sempre consulta os logs antes de tentar resolver um problema. Isso permite tomar uma ação mais informada e eficaz. O `rsyslog` é uma ferramenta de monitoramento de logs padrão em muitas distribuições Linux. Ele pode ser usado para coletar logs de várias fontes e armazená-los em um local central.

O `rsyslog` usa um modelo cliente-servidor. O cliente e o servidor podem existir no mesmo *host* ou em máquinas diferentes. As mensagens são enviadas e recebidas em um formato específico e podem ser mantidas em servidores `rsyslog` centralizados em redes IP. O *daemon* do `rsyslog` – o `rsyslogd` – trabalha com o `klogd` (que gerencia as mensagens do kernel).

> **Nota:** Um *daemon* é um serviço executado em segundo plano. Observe o "d" final nos nomes dos *daemons*: `klogd` ou `rsyslogd`.

10.1 Tipos de log

Como os logs são dados variáveis, costumam ser encontrados em `/var/log`. Vamos conhecer alguns destes logs:

- **`/var/log/auth.log`** : atividades relacionadas aos processos de autenticação: usuários registrados, informações de `sudo`, *cron jobs*, tentativas de login malsucedidas etc.;
- **`/var/log/syslog`** : um arquivo centralizado para praticamente todos os logs capturados pelo `rsyslogd`. Por incluir muitas informações, os logs são distribuídos por outros arquivos de acordo com a configuração fornecida em `/etc/rsyslog.conf` ;
- **`/var/log/debug`** : informação de depuração dos programas;
- **`/var/log/kern.log`** : mensagens do kernel;
- **`/var/log/messages`** : mensagens informativas que não estão relacionadas ao kernel, mas a outros serviços. É também o destino padrão do log do cliente remoto em uma implementação de servidor de log centralizado;

- `/var/log/daemon.log` : informações relacionadas aos daemons ou serviços em execução em segundo plano;
- `/var/log/mail.log` : informações relacionadas ao servidor de e-mail, como o Postfix.

Somente o usuário administrador (*root*) ou usuário com permissão equivalente pode ler os arquivos de log. Você pode usar um paginador como `less` , por exemplo:

```
# less /var/log/auth.log
```

Outra maneira de se ler arquivo de log é usando o comando `tail` . Com ele, é possível visualizar as últimas 10 linhas do log e mantê-lo monitorado:

```
# tail -f /var/log/auth.log
```

10.2 Instalando um servidor de log

A necessidade de registro das atividades dos usuários e serviços dos sistemas é notoriamente importante. A norma NBR ISO/IEC 27002 recomenda, no item 10.10.1, as seguintes características de um sistema de logs:

- Identificação do usuário;
- Datas e horários de entrada e saída de terminais;
- Hostname ou endereço IP, para serviços acessados via rede;
- Registro das tentativas de acesso aceitas e rejeitadas.

No exame, você precisa compreender o fundamento de um serviço de log. Vamos usar como exemplo o servidor `rsyslog` . Faça sua instalação com o seguinte comando:

```
# sudo apt-get install rsyslog
```

O arquivo de configuração encontra-se em `/etc/rsyslog.conf` e sua forma de configuração é similar ao `syslog`. Cada linha do arquivo é organizada pela seguinte sintaxe:

```
facilidade.nivel destino
```

Definição	Explicação
Facilidade	Especifica que tipo de programa está enviando a mensagem
Nível	Especifica o nível de gravidade da mensagem
Destino	Especifica para onde deve ser encaminhada a mensagem de log

Tipos de facilidades:

Tipo	Explicação
auth	Mensagens de segurança/autorização - Tentativas de login
authpriv	Mensagens de segurança/autorização (privadas)
cron	Serviço de agendamento do cron (`cron` e `at`)
daemon	Outros serviços do sistema que não possuem facilidades específicas
ftp	Serviços de FTP do sistema
kern	Mensagens do kernel
lpr	Subsistema de impressão
Local0-7	Reservados para uso local
mail	Subsistema de e-mail
news~	Subsistema de notícias da USENET
security	Sinônimo para a facilidade `auth`
user	Mensagens genéricas de nível de usuário
*	Confere todas as facilidades

Tipos de níveis:

Níveis	Definição
emerg	O sistema está inutilizável
alert	Uma ação deve ser tomada imediatamente
crit	Condições críticas
err	Condições de erro
warning	Condições de alerta
notice	Condição normal, mas significante
info	Mensagens informativas
debug	Mensagens de depuração
*	Todos os níveis

Tipos de destinos:

Tipos	Definição
arquivo	O Rsyslog enviará os logs para um arquivo. Opção mais comum.
|	O Rsyslog enviará os logs através de um *pipe*. Muito usado para redirecionar logs a um terminal.
@	O Rsyslog enviará seus logs para um computador remoto, utilizando IP ou hostname.
user1,user2	Especificando o usuário, o log enviará as mensagens para o usuário especificado. Pode-se separar os usuários por vírgula.
*	O log será enviado para todos os usuários logados no sistema, por meio do comando "wall".

10.3 Configurando o sistema de logs no cliente

Instale o pacote do `rsyslog` no cliente:

```
# apt-get install rsyslog
```

Vamos criar um arquivo de log com todas as facilidades, ou seja, com todos os logs disponíveis. Abra o arquivo de configuração em `/etc/rsyslog.conf` e insira estas linhas:

```
*.* /var/log/tudo.log
*.* @192.168.10.11
```

Na primeira linha, estamos redirecionando todos os logs para um arquivo em `/var/log/tudo.log` e, na segunda, redirecionamos todos os logs para o servidor (que vamos criar) cujo IP é 192.168.10.11.

Reinicie o `rsyslog` :

```
# /etc/init.d/rsyslog stop
# /etc/init.d/rsyslog start
```

Faça algumas tentativas de login e observe o arquivo `/var/log/tudo.log` .

10.4 Configurando o servidor de log

Primeiro, é necessário que o servidor seja habilitado para receber logs de outras máquinas. Para isso, acrescente o parâmetro `-r` :

```
# vim /etc/default/rsyslog
```

Coloque a linha:

```
RSYSLOGD_OPTIONS="-c4,-r"
```

Depois precisamos remover o comentário no arquivo `/etc/rsyslog.conf` das linhas:

```
$ModLoad imudp
$UDPServerRun 514
```

Crie também uma entrada para os logs que você vai receber:

```
*.* /var/log/todos_logs.log
```

Reinicie o serviço do `rsyslog`. Instale o pacote do `rsyslog` no cliente:

```
# apt-get install rsyslog
```

Vamos criar um arquivo de log com todas as facilidades, ou seja, com todos os logs disponíveis. Abra o arquivo de configuração em `/etc/rsyslog.conf` e insira estas linhas:

```
*.* /var/log/tudo.log
*.* @192.168.10.11
```

Na primeira linha, estamos redirecionando todos os logs para um arquivo em `/var/log/tudo.log` e, na segunda, redirecionamos todos os logs para o servidor (que vamos criar) cujo IP é 192.168.10.11.

Reinicie o `rsyslog`:

```
# /etc/init.d/rsyslog stop
# /etc/init.d/rsyslog start
```

Faça algumas tentativas de login e observe o arquivo `/var/log/tudo.log`.

Conclusão

O uso de logs no sistema operacional Linux é de extrema importância para a gestão, manutenção e segurança do ambiente computacional. Os logs fornecem informações detalhadas sobre eventos, atividades e desempenho do sistema, permitindo a administradores e usuários identificarem problemas, diagnosticarem falhas, otimizarem recursos e monitorarem a segurança. O exame concentra-se nos locais dos arquivos mais comuns de log e também no seu conhecimento sobre as facilidades e os níveis.

Capítulo 11
Tópicos 108.3 e 108.4

11.1 108.3 – Fundamentos de MTA (Mail Transfer Agent)

Certificação	LPIC-1
Versão	5.0
Tópico no exame	108.3 – Fundamentos de MTA (*Mail Transfer Agent*)
Objetivos	Os candidatos devem estar cientes dos programas MTA comumente usados e devem ser capazes de realizar as configurações básicas dos arquivos `/etc/aliases` e `.forward` em um computador cliente. Outros arquivos de configuração não são cobrados.
Peso	3

Este é, sem dúvida, um dos menores capítulos do livro. A certificação LPIC-1 exame 102 exige apenas um conhecimento superficial sobre ***Mail Transfer Agent*** (MTA), que é o programa responsável por controlar o envio e o recebimento de mensagens de correio eletrônico de forma local ou remota.

Há várias opções de MTAs, sendo os mais conhecidos o Sendmail, o Postfix, o Qmail e o Exim. O MTA roda como um serviço do sistema, geralmente utilizando a porta 25, responsável pelo protocolo SMTP.

Em ambientes UNIX, é possível interagir com o funcionamento do MTA de diversas formas. Entre as mais comuns está criar um redirecionamento de e-mail. O redirecionamento pode ser definido

no escopo geral do sistema ou pelo próprio usuário. Para definir um redirecionamento geral do sistema, é utilizado o arquivo `/etc/aliases`. Nele, é possível vincular nomes diferentes - conhecidos como *aliases* - para um ou mais destinatários do sistema. Exemplo de um `/etc/aliases`:

```
gerente: root
contato: root
webmaster: root
```

Nesse exemplo, as mensagens encaminhadas para os usuários `gerente`, `contato` e `webmaster` serão redirecionadas para o usuário `root`.

Após fazer alterações nesse arquivo, é necessário executar o comando:

```
# newaliases
```

Também é possível fazer com que o MTA encaminhe as mensagens recebidas para outro usuário ou endereço de e-mail, editando o arquivo `~/.forward` no diretório pessoal. Esse arquivo pode conter um ou mais endereços para os quais os e-mails recebidos pelo usuário em questão serão direcionados. Para exibir a fila de e-mail e o estado das mensagens que estão sendo transferidas, utiliza-se o comando `mailq`.

Para enviar um e-mail podemos utilizar o comando a seguir. Este exemplo considera a instalação do pacote `mailutils` do Debian e envia uma mensagem para o usuário conectado.

```
echo "Minha mensagem de e-mail para o usuário dessa máquina." | mail -s
"Título da mensagem" $USER@localhost
```

Para checar o arquivo com o conteúdo, use:

```
cat /var/mail/$USER
```

Para enviar e-mails externos, você precisa configurar as credenciais de acesso do seu servidor de e-mail.

Como este conteúdo é de fato bem pequeno, vamos aproveitar para cobrir neste capítulo outro tópico cobrado e que também tem peso 1 no exame, que é o gerenciamento de impressoras e impressão.

11.2 Tópico 108.4 – Configurar impressoras e impressão

Certificação	LPIC-1
Versão	5.0
Tópico no exame	108.4 – Configurar impressoras e impressão
Objetivo	Os candidatos devem ser capazes de gerenciar filas de impressão e trabalhos de impressão do usuário utilizando o CUPS e a interface de compatibilidade LPD.
Peso	2

Mesmo com diversos documentos digitais, haverá momentos em que você poderá necessitar realizar a impressão de um documento. No Linux, o programa responsável pelo sistema de impressão se chama **CUPS** (*Common Unix Printing System*). O CUPS fornece controle sobre impressoras, filas de impressão, impressão remota e compatibilidade com as ferramentas do sistema de impressão antigo, chamado de *LPD*.

O modo mais fácil de utilizar o CUPS é através da interface web, mas antes você deve instalá-lo. Para isso, utilize o seguinte comando:

```
# sudo apt-get install cups
```

Após a instalação, para abrir a página de gerenciamento do CUPS, você deve abrir o seu navegador web e colocar o endereço:

`http://localhost:631`

Adicionar impressora

Para adicionar uma impressora, após abrir a página web de gerenciamento do CUPS, clique em `Administration - Add Printer` e, em seguida, logue-se como `root`. Escolha o seu protocolo de impressão no menu. As opções mais comuns são *local* (do tipo USB) ou *remota*.

Após selecionar o seu tipo de conexão, insira o nome da impressora (exemplo: "HP-Deskjet"), insira uma descrição para a sua impressora (exemplo: "Setor de marketing - Sala 03"), selecione o fabricante da impressora, o modelo e finalize clicando em `Add Printer`.

Ver os jobs de impressão

Para verificar os trabalhos enviados à impressora, clique no menu `Jobs`. Nessa seção, você pode cancelar ou visualizar os trabalhos de impressão.

Cancelar todos os trabalhos de impressão

Para cancelar todos os trabalhos de impressão, clique em `Printers`, selecione a sua impressora e, na guia `Maintenance`, selecione a opção `Cancel All Jobs`.

Resolução de problemas de impressão

Resolver problemas de impressão no Linux pode envolver várias etapas, dependendo da natureza específica do problema que você está enfrentando. A seguir, deixo alguns passos essenciais sobre este tema.

- **Verifique a conexão da impressora:** Certifique-se de que a impressora está devidamente conectada ao computador e ligada. Verifique se os cabos USB ou de rede estão corretamente conectados.

- **Verifique se a impressora foi reconhecida:** Abra um terminal e execute o comando `lpstat -a`. Isso mostrará todas as impressoras disponíveis. Certifique-se de que sua impressora esteja listada.
- **Verifique os trabalhos de impressão:** Execute o comando `lpq` para verificar se há trabalhos de impressão na fila. Você pode cancelar trabalhos usando o comando `lprm`.
- **Reinicie o sistema de impressão:** O comando para reiniciar o sistema de impressão pode variar dependendo da distribuição, mas geralmente é algo como:

```
# sudo systemctl restart cups
```

- **Verifique os drivers da impressora:** Certifique-se de que os drivers da impressora estão instalados corretamente. Algumas distribuições do Linux podem ter ferramentas gráficas para instalar drivers de impressoras. Verifique também se o *firewall* não está bloqueando a comunicação com a impressora. Você pode temporariamente desativar o *firewall* para testar.
- **Atualize os drivers:** Certifique-se de ter os drivers mais recentes para a sua impressora. Muitas vezes, os fabricantes fornecem drivers específicos para Linux.
- **Teste com outro aplicativo:** Tente realizar a impressão a partir de diferentes aplicativos, para garantir que o problema não seja específico de um programa.
- **Verifique configurações de papel e qualidade:** Certifique-se de que as configurações de papel e qualidade de impressão estejam corretas.
- **Consulte a comunidade ou o fórum da distribuição:** Caso as etapas anteriores não resolvam o problema, consulte a comunidade de usuários da sua distribuição Linux. Muitas vezes,

outros usuários podem ter enfrentado problemas semelhantes e podem fornecer orientações específicas.

Lembre-se de que a resolução de problemas de impressão pode variar dependendo da distribuição Linux específica que você está usando. Recomendo consultar a documentação da sua distribuição e os recursos de suporte específicos.

Conclusão

Os tópicos 108.3 e 108.4 possuem peso 1 no exame e exigem apenas um conhecimento superficial dos temas. No caso do tópico 108.3, sobre MTA, foque sua atenção para o arquivo `/etc/aliases` e o que ele faz (redirecionamento de e-mails). Já em relação ao tópico 108.4, sobre gerenciamento de impressão, lembre-se do nome do servidor CUPS e do seu local padrão de funcionamento (`http://localhost:631`).

Tópico 109: Fundamentos de rede

Capítulo 12
Tópico 109 – Fundamentos de rede

Certificação	LPIC-1
Versão	5.0
Tópicos no exame	109.1 – Fundamentos de protocolos de internet; 109.2 – Configuração persistente de rede; 109.3 – Soluções para problemas simples de rede; e 109.4 – Configurar DNS cliente
Objetivo	Os candidatos devem ser capazes de: 1) demonstrar um conhecimento adequado sobre os fundamentos das redes TCP/IP; 2) administrar a configuração persistente de rede em um sistema Linux; 3) solucionar problemas de rede em computadores clientes; e 4) configurar o DNS em um computador cliente.
Peso	14

12.1 109.1 Fundamentos de protocolos de internet

Este tópico cobre os fundamentos básicos sobre TCP/IP. Mas, afinal, o que é o TCP/IP?

O TCP/IP (do inglês, *Transmission Control Protocol/Internet Protocol*, ou Protocolo de Controle de Transmissão/Protocolo de Internet) é um conjunto de protocolos de comunicação utilizados para conectar dispositivos em uma rede, incluindo a internet. É uma arquitetura de comunicação que define como os dados devem ser transmitidos, roteados e recebidos entre dispositivos em redes. A arquitetura TCP/IP é composta por dois principais protocolos:

TCP (Protocolo de Controle de Transmissão - Transmission Control Protocol):

- É responsável pelo controle de transmissão de dados, garantindo a entrega ordenada e confiável dos pacotes;
- Estabelece e mantém a conexão entre os dispositivos, garantindo que os dados sejam entregues sem erros, na ordem correta e sem duplicatas.
- Evita congestionamentos e controla o fluxo das informações

IP (Protocolo de Internet - Internet Protocol):

- É responsável pelo roteamento dos pacotes de dados através da rede;
- Atribui endereços IP únicos a cada dispositivo conectado à rede, permitindo a identificação e localização dos dispositivos na rede.
- Estabelece comunicações de controle entre roteadores e sistemas finais (ICMP)

Esses protocolos são essenciais para a comunicação de dados na internet e em redes locais. O TCP/IP é composto por várias camadas, sendo as principais:

- **Camada de aplicação:** fornece serviços de aplicação diretamente aos aplicativos do usuário, como HTTP, SMTP, FTP etc.;
- **Camada de transporte:** garante a entrega confiável dos dados, sendo o TCP um exemplo comum nesta camada;
- **Camada de internet:** lida com o endereçamento, o roteamento e o encaminhamento de dados, utilizando o

protocolo IP (IPv4 ou IPv6);
- **Camada de interface de rede:** lida com os detalhes da conexão física entre os dispositivos e a rede.

A arquitetura TCP/IP é uma base fundamental para a comunicação eficaz entre diferentes sistemas e é amplamente adotada na internet e em muitas redes corporativas e domésticas, permitindo a comunicação entre dispositivos heterogêneos de maneira padronizada e eficiente.

12.2 O endereço IPv4

O endereço IP (IPv4) é formado por 32 bits, separados em 4 grupos de 8 bits, chamados comumente de *octetos*, representados de maneira decimal:

- **192.168.10.20** = 11000000.10101000.00001010.00010100
- **0.0.0.0** = 00000000.00000000.00000000.00000000
- **255.255.255.255** = 11111111.11111111.11111111.11111111

O endereço IP é separado por classes. São elas:

Classe	Primeiro Octeto	Range
A	1-126	1.0.0.0-126.255.255.255
B	128-191	128.0.0.0-191.255.255.255
C	192-223	192.0.0.0-223.255.255.255

Exemplos:

- Classe A: **10.25.13.10; 110.14.20.12**
- Classe B: **141.150.200.1; 172.168.1.1**
- Classe C: **192.168.20.30; 200.178.12.242**

As classes são definidas pelo *range* do primeiro octeto. Observe que, da classe A para a B, pulamos o número 127, que é reservado para o endereço de *loopback*, ou seja, sua própria máquina (127.0.0.1).

Endereçamento IP - Privado vs. Público

Todas as combinações possíveis de 32 bits que formam um endereço IPv4 totalizam 4.294.967.296 possibilidades (ou pouco mais de 4 bilhões, para ser mais claro). Caso todas as máquinas do mundo utilizassem um IP único (incluindo impressoras, câmeras, computadores, celulares etc.), não teríamos endereços suficientes. Para resolver essa questão matemática, temos faixas de IP privadas. Quando conectamos em nosso provedor, temos um IP público (válido para toda a internet), o qual pode ser consultado em diversos sites (por exemplo, o site www.meuip.com.br).

Os endereços privados podem ser repetidos em redes particulares, desde residências até grandes empresas, e são sempre de uma das três faixas a seguir:

- **Classe A**: de 10.0.0.0 a 10.255.255.255
- **Classe B**: de 172.16.0.0 a 172.31.255.255
- **Classe C**: de 192.168.0.0 a 192.168.255.255

Máscara de rede

A máscara de rede é um número que usa o mesmo formato do IP e é utilizada para definir qual parte do IP representa a rede e qual parte representa o *host* (dispositivo). Ela também é composta de 32 bits em 4 grupos de 8:

- **255.0.0.0** = 11111111.00000000.00000000.00000000 = 8
- **255.255.0.0** = 11111111.11111111.00000000.00000000 = 16
- **255.255.255.0** = 11111111.11111111.11111111.00000000 = 24

- **255.255.255.255** = 11111111.11111111.11111111.11111111 = 32

Os primeiros números do octeto representam a rede e os demais, os *hosts*. Se observar a máscara da **classe C (255.255.255.0)**, perceberá que podemos usar 254 IPs para *hosts*, já que um (255) será exclusivo para o endereço de *broadcast*, que determina o final da rede (envio para todos os *hosts*).

As máscaras padrão são:

Classe	Primeiro Octeto	Range	Máscara Padrão
A	1-126	1.0.0.0-126.255.255.255	255.0.0.0/8
B	128-191	128.0.0.0-191.255.255.255	255.255.0.0/16
C	192-223	192.0.0.0-223.255.255.255	255.255.255.0/24

Exemplos:

- Classe A: **110**.14.20.12
- Classe B: **172.168**.1.1
- Classe C: **200.178.12**.242

Veja outros exemplos sobre limitar ou reservar IPs:

- **Exemplo 1: 192.168.8.12 / 255.255.255.0 / 24**
    - Range: 192.168.8.0 - 192.168.8.255
    - Endereço de rede: 192.168.8.0
    - Endereço de broadcast: 192.168.8.255
    - Hosts: 192.168.8.1 - 192.168.8.254
- **Exemplo 2: 192.168.8.12 / 255.255.0.0 / 16**

- Range: 192.168.0.0 - 192.168.255.255
- Endereço de rede: 192.168.0.0
- Endereço de broadcast: 192.168.255.255
- Hosts: 192.168.0.1 - 192.168.255.254

No exemplo 1, caso tenha uma máquina com IP **192.168.10.10**, ele não pertence à rede, já que o terceiro byte deste IP é diferente do terceiro byte do IP de referência. Para uma máscara de 24 bits como a do exemplo 1, os três primeiros bytes precisam ser iguais para estarem na mesma rede. Já este mesmo IP no exemplo 2 pertenceria à rede porque, nesse caso, a máscara de 16 bits exige apenas que os 2 primeiros bytes sejam iguais.

Rota padrão

A rede 192.168.10.1 não consegue enxergar a rede 192.168.200.1 porque estão em sub-redes diferentes. Para que essas redes possam se comunicar, elas precisam de um intermediário, chamado **gateway**.

Quando um host verifica que o endereço de destino não pertence à sua rede, ele envia o pacote para o endereço do gateway de sua própria configuração. O gateway precisa estar conectado na mesma rede do remetente e a outras redes simultaneamente. Há nele uma configuração de rotas de modo que ele seja o equipamento capaz de fazer com que a informação "navegue" entre as sub-redes. Geralmente um roteador ou um servidor com várias interfaces de rede realizam essa atividade. E após passar por um conjunto de roteadores a informação chega ao seu destino.

12.3 Padrão IPv6

O IPv6 é um novo padrão de endereçamento IP. Cada endereço possui 128 bits, divididos em 8 grupos de 16 bits, representados por

4 números hexadecimais. Por exemplo: `2001:0db8:85a3:08d3:1319:8a2e:0370:7344`. Veja que, mesmo em hexadecimal, o valor é bem grande — caso fossem decimais, seriam 16 dígitos em cada grupo! Embora o IPv4 possa fornecer mais de 4 bilhões de endereços, isso não é suficiente hoje em dia. Com o avanço da "Internet das Coisas", o IPv6 vem sendo incorporado aos computadores e a maioria dos dispositivos consegue trabalhar com os dois tipos de endereços. O IPv4 ainda é, de longe, o mais utilizado, mas essa realidade pode mudar com o passar dos anos.

No IPv6, temos três tipos de endereço:

- **Unicast:** Identifica uma única interface;
- **Multicast:** Identifica um conjunto de interfaces. Um pacote é enviado para cada interface;
- **Anycast:** Identifica um conjunto de interfaces. Um pacote é enviado para uma única interface, normalmente a mais próxima.

O Unicast representa um host específico. O Multicast é semelhante ao Broadcast: ele identifica um conjunto de interfaces de rede e um pacote encaminhado para Multicast é enviado para essas interfaces. O Anycast identifica um conjunto de interfaces da rede, mas o pacote é enviado para a interface mais próxima.

Abreviações

TCP (Transmission Control Protocol)

- O protocolo TCP é o protocolo que garante que o pacote enviado do *host* A para o *host* B seja entregue corretamente;
- O TCP também garante que os pacotes cheguem na ordem correta;
- O TCP é utilizado por serviços como FTP, SMTP, HTTP etc.

UDP (User Datagram Protocol)

- O pacote UDP é um protocolo de transporte não orientado a conexão;

- O UDP não garante a entrega dos pacotes;
- Melhor performance na transmissão de dados;
- Utilizado por serviços como DNS, NFS e NTP.

ICMP (Internet Control Message Protocol)

O ICMP é um protocolo usado para transmitir informações de controle entre os elementos da rede:

- Controle de volume de tráfego;
- Detecção de destinos não atingíveis;
- Redirecionamento de rotas;
- Verificação de status de *hosts* remotos;
- Utilizado em comandos como o `ping` e `traceroute`.

Portas e serviços

- Cada serviço é provido através de uma porta;
- Portas possuem 16 bits, ou seja, um máximo de 65535;
- As primeiras 1024 portas são reservadas a serviços conhecidos;
- As demais são utilizadas para serviços específicos ou para a porta de *socket* de uma conexão.

Exemplos:

- 200.20.125.60:25 (Serviço SMTP);
- 172.168.10:80 (Serviço HTTP);
- `[2001:0db8:85a3:0000:0000:1319:0000:7344]` : 80.

Veja na tabela abaixo as principais portas e serviços:

Portas	Serviço
20 e 21	FTP
22	SSH
23	Telnet
25 e 465	SMTP e SMTPS
53	DNS
67 e 68	DHCP
80 e 443	HTTP e HTTPS
110 e 995	POP3 e POP3S
123	NTP
139	Netbios
143 e 993	IMAP e IMAPS
161 e 162	SNMP
389 e 636	LDAP e LDAPS

12.4 109.2 Configuração persistente de rede

O primeiro passo é, no VirtualBox, habilitar uma segunda placa de rede para nossos testes. Clique em *Configurações* - *Rede* - *Adaptador 2* - *Habilitar placa de rede*. Também deixe as duas placas no modo `bridge`.

Quando iniciar o sistema, caso esteja usando a interface gráfica, é importante desabilitar o NetworkManager (que faz as configurações

de rede no modo gráfico). Para isto, execute:

```
# systemctl stop NetworkManager
```

Para desabilitar o NetworkManager no *boot* da máquina, de modo que prevalecerá nossa configuração manual, execute:

```
# systemctl disable NetworkManager
```

Os arquivos básicos de configuração são:

- `/etc/services` : mostra todas as portas e serviços;
- `/etc/hostname` : permite definir o nome de sua máquina;
- `/etc/hosts` : permite mapear um IP para um nome;
- `/etc/networks` : permite mapear o nome de redes. Exemplo: `rede-remota 192.168.10.0` ;
- `/etc/nsswitch.conf` : indica onde o sistema procura as informações de rede e de autenticação;
- `/etc/resolv.conf` : este arquivo deve ter o nome do servidor DNS a ser utilizado.

Se você executar o comando `ls -l` no arquivo `/etc/resolv.conf` , você vai observar que o NetworkManager cria um link simbólico para `../run/resolvconf/resolv.conf` , e isso muda o padrão do Linux. Toda vez que você reiniciar a máquina, vai perder o que escrever em `/etc/resolv.conf` . Para resolver isso, remova o arquivo e crie-o novamente:

```
# rm -f /etc/resolv.conf
# touch /etc/resolv.conf
```

Agora que você criou o novo arquivo, coloque o endereço do seu DNS usando o padrão:

```
nameserver 192.168.15.1
```

Algumas pessoas utilizam o DNS do Google: 8.8.8.8.

Comandos básicos de rede

O comando mais básico e conhecido é o `ifconfig` . Esse comando mostra as interfaces de rede disponíveis. Nas distribuições atuais, esse comando pode não estar presente. Neste caso, você pode fazer a instalação do pacote `net-tools` :

```
# apt-get install net-tools
```

Quando você digita o comando sem argumentos, ele exibe o nome e a configuração de sua interface de rede:

```
# ifconfig
```

Caso sua interface de rede não esteja ativa, você pode visualizar sua configuração com o seguinte comando (a opção `-a` equivale a *all*, ou *todas as interfaces*):

```
# ifconfig -a
```

Se você deseja obter informação de uma interface específica, digite:

```
# ifconfig enp0s3
```

O nome `enp0s3` pode variar de acordo com sua distribuição de Linux. Durante muitos anos o padrão utilizado era `eth0` para a primeira interface, mas isso hoje em dia não existe mais, ou seja, o nome pode mudar de acordo com sua distribuição.

Para derrubar uma interface de rede, ou seja, desabilitá-la, utilize:

```
# ifconfig enp0s3 down
```

Para subir (habilitar) a interface, utilize:

```
# ifconfig enp0s3 up
```

Para definir um endereço de IP específico, utilize:

```
# ifconfig enp0s3 192.168.15.100
```

Quando você não informa a máscara de rede, ele utiliza a máscara padrão da classe do IP. Se você deseja inserir uma máscara

manualmente, digite:

```
# ifconfig enp0s3 down
# ifconfig enp0s3 192.168.15.100 netmask 255.255.255.10 up
```

> O `ifconfig` está cada vez mais em desuso, e versões atuais dos sistemas não permitem mais o uso do `ifconfig` ou de algumas de suas funções.

Comando `route`

O comando `route` mostra as informações de rota (*gateway*). O modo mais simples de usar esse comando é (a opção `-n` mostra sempre endereços numéricos em vez de tentar substituir endereços por nomes):

```
# route -n
```

Para adicionar uma rota padrão (endereço do *gateway*), use o comando `add` , da seguinte forma:

```
# route add default gw 192.168.15.2
```

Para remover uma rota padrão (`default gw`), use o comando `del` , da seguinte forma:

```
# route del default gw 192.168.15.2
```

Subindo sem ifconfig

Caso você visualize com o comando acima que sua interface está derrubada (`down`), você pode torná-la ativa sem o uso do `ifconfig` . Para isso, utilize o seguinte comando:

```
# ip link set eth1 up
```

A interface está ativa, mas ainda não tem IP. Para incluir um endereço IP, utilize:

```
# ip addr add 192.168.1.55/24 dev eth1
```

Verifique se o IP foi adicionado:

```
# ip addr show
```

Na distribuição Linux *Centos* (Configuração estática):

```
# vi /etc/sysconfig/network-scripts/ifcfg-eth1

DEVICE=eth1
BOOTPROTO=none ou static
IPADDR=192.168.1.15
NETMASK=255.255.255.0
GATEWAY=192.168.1.1
ONBOOT=yes
```

Para reiniciar o serviço de rede, utilize:

```
# service network restart
```

12.5 109.3 Soluções para problemas simples de rede

Na seção anterior, estudamos o comando `ifconfig`. Nesta seção, estudaremos outro comando comum e que é cobrado neste tópico no exame LPIC 102-500. É o comando **`ping`**, utilizado para enviar pacotes ICMP para um host e verificar a latência da rede, uma vez que ele envia o pacote e espera a resposta, contabilizando esta ação. Para que o comando `ping` funcione, o firewall do destino tem que estar configurado para aceitar pacotes ICMP.

Veja alguns exemplos de uso do comando `ping`:

```
# ping www.google.com.br
# ping 192.168.15.1
```

Para formatos IPv6, usamos o comando `ping6`, uma vez que o `ping` funciona apenas para IPv4.

```
# ping6 ipv6.google.com
```

Seu roteador deve aceitar IPv6.

Comando `traceroute`

O comando `traceroute` vai mostrar todo o caminho (rota) de um host a outro. Por exemplo:

```
# traceroute -I www.uol.com.br
```

A opção `-I` vai fazer o `traceroute` usar ICMP em vez de UDP, que é o padrão. Para IPv6, temos o `traceroute6`. O comando `tracepath` é similar ao `traceroute`.

Comando `host`

O DNS é um servidor de nomes que converte *host* em IP e IP em *host*. O arquivo em que definimos o DNS é o `/etc/resolv.conf`. Para verificar o IP de um *host*, podemos utilizar o seguinte comando:

```
# host www.debian.com
```

Podemos também fazer o DNS reverso:

```
# host 200.17.202.197
```

Comando `nslookup`

O comando `nslookup` faz praticamente a mesma coisa do `host`, mas exibe mais detalhes:

```
# nslookup debian.org
```

Por exemplo, para saber qual é o servidor de e-mail do `debian.org`, o seguinte comando poderia ser utilizado:

```
# nslookup -query=MX debian.org
```

Para retornar todas as informações:

```
# nslookup -query=any debian.org
```

Comando `dig`

O comando `dig` hoje está mais em uso do que o `nslookup`, pois ele é bem detalhado.

```
# dig ubuntu.com
```

Para ver, por exemplo, informação do servidor de e-mail:

```
# dig ubuntu.com mx
```

Para fazer o DNS reverso:

```
# dig -x 91.189.89.110
```

Identificar problema de rota

Para ver nossa tabela de `route`, usamos o comando:

```
# route -n
```

Por exemplo, quando você executa o comando `ping www.uol.com.br`, o sistema vai primeiramente procurar dentro da sua rede (não precisa do gateway); como ele não encontra, ele joga para o gateway padrão. Se você não tem um, não conseguirá acessar a internet, por exemplo:

```
# route del default
```

Agora que removemos, caso esteja usando um DNS externo, execute:

```
# ping 8.8.8.8
```

E observará que a rede está inacessível. Para adicionar novamente a rota:

```
# route add default gw 192.168.15.1
```

Identificar problema de porta

O comando `netstat` vai demonstrar todas as portas abertas na máquina. As portas abertas estão associadas a serviços de rede executando nessa máquina aguardando por requisições. Exemplo:

```
# netstat -nalpt
```

Destrinchando a opção que usamos (`nalpt`): `n` - não resolve endereços ou portas (mostra números); `a` - mostrar tudo; `p` - os processos; `t` - conexões TCP ativas; `l` - modo de escuta.

Quando estiver com dúvida sobre alguma porta, pode pesquisar no `/etc/services` . Por exemplo:

```
# grep 25 /etc/services
```

O `netstat` também permite ver a rota da máquina, de forma semelhante ao comando `route -n` :

```
# netstat -rn
```

O comando `netcat`

Podemos usar o comando `nc` (abreviação do comando `netcat`) para ficar ouvindo uma porta (abrir uma porta). Exemplo:

```
# nc -l -p 12345
```

Agora observe esta porta:

```
# netstat -nalpt | grep 12345
```

Vendo que a porta está ouvindo, vamos abrir o `telnet` nesta porta:

```
# telnet 127.0.0.1 12345
```

O comando `tcpdump`

O comando `tcpdump` tem muitas opções, mas, para a prova, precisamos saber apenas o que ele faz. O `tcpdump` serve para monitorar os pacotes que estão circulando no host. Por exemplo:

```
# tcpdump host 192.168.15.1
```

12.6 109.4 Configurar DNS cliente

Nós, seres humanos, somos familiarizados com nomes. Para nós, é fácil entender o que significa `www.casadocodigo.com.br`, mas é bem complicado entender que o endereço IP que leva até a esse destino é `23.227.38.32`. Nós precisamos dos nomes para reconhecer os destinos, e os computadores usam os números para decidir as melhores rotas. Como podemos resolver isso? Com um serviço chamado Domain Name System (DNS), que converte nomes em endereços (e vice-versa).

Os clientes DNS usam um pacote UDP para enviar o nome para um servidor DNS e recebem um endereço IP como resposta. Assim, o cliente pode usar o IP para estabelecer uma conexão TCP com o site de destino. Os arquivos principais de configuração do cliente DNS são:

- `/etc/resolv.conf`
- `/etc/hosts`
- `/etc/nsswitch.conf`

Os comandos principais:

- `host`
- `dig`
- `nslookup`

Temos um outro comando chamado `getent`, que normalmente é utilizado para consultar usuários. Por exemplo:

```
# getent passwd juliano
```

No caso de DNS, podemos usar desta forma:

```
# getent hosts www.uol.com.br
```

Ele retornou IPv6. Para IPv4, fazemos assim:

```
# getent ahostsv4 www.uol.com.br
```

O arquivo `nsswitch.conf`

Este arquivo coloca a ordem para a resolução de nomes. Se você observar, ele define, em `hosts`, que o sistema deve procurar primeiro em *files* (arquivos) e depois no DNS.

Neste caso, se colocarmos, por exemplo, um IP qualquer em `/etc/hosts`, ele vai usar isso antes de procurar no DNS. Veja:

```
# nano /etc/hosts
200.200.200.200 lpi.org
```

Agora, ao realizar o comando `ping lpi.org`, ele vai resolver como se o IP fosse `200.200.200.200`! Se você inverter a ordem no `/etc/nsswitch` colocando-o para ler primeiro o DNS, o IP correto do `lpi.org` vai funcionar, porque o arquivo `/etc/hosts` será a segunda pesquisa, caso a primeira falhe.

Conclusão

A importância de aprender os fundamentos de redes para a prova LPIC 102-500 é significativa, já que este é um dos tópicos de maior peso no exame. Além disso, um(a) profissional administrador(a) de sistemas Linux atuará diariamente com os conceitos aqui apresentados:

- **Administração de sistemas:** como administrador(a) de sistemas Linux, você frequentemente precisará configurar, manter e solucionar problemas em redes;
- **Conectividade:** os sistemas Linux são amplamente usados em ambientes de rede, desde servidores até dispositivos incorporados. Ter conhecimento sobre redes ajuda a garantir que esses sistemas possam se comunicar de forma eficaz;
- **Segurança:** a segurança de redes é uma parte crítica da administração de sistemas Linux. Conhecer os fundamentos de redes permite que você configure firewalls, monitore o tráfego de rede e implemente medidas de segurança para proteger os sistemas e os dados;
- **Solução de problemas:** quando ocorrem problemas de conectividade, latência ou perda de pacotes, é fundamental compreender os conceitos de redes para diagnosticar e resolver esses problemas de maneira eficaz;
- **Gerenciamento de servidores:** em ambientes corporativos, os servidores Linux geralmente desempenham um papel fundamental. Ter uma compreensão sólida dos fundamentos de redes é necessário para configurar e gerenciar esses servidores;
- **Desenvolvimento profissional:** Aprender os fundamentos de redes não é apenas importante para a certificação LPIC, mas também para a sua carreira em TI como um todo. É uma habilidade valiosa que pode abrir portas para várias oportunidades profissionais.

Em resumo, os fundamentos de redes desempenham um papel crucial na administração de sistemas Linux e são um componente fundamental da certificação LPIC-1.

Tópico 110: Segurança

Capítulo 13
Tópico 110.1 - Tarefas administrativas de segurança

Certificação	LPIC-1
Versão	5.0
Tópico no exame	110.1 - Tarefas administrativas de segurança
Objetivo	Os candidatos devem ser capazes de examinar a configuração do sistema para garantir a segurança do computador, de acordo com as políticas de segurança local
Peso	3

A certificação LPIC-1 102 começou a exigir dos candidatos um conhecimento prévio em segurança da informação. Como administrador de sistemas, você deve ter conhecimento de que os servidores corporativos são frequentemente alvo de ataques por parte de *crackers* e bots maliciosos. Reforçar a segurança ajuda a prevenir acessos não autorizados, evitando que invasores obtenham controle sobre o sistema. Além disso, é necessário ter uma atenção sobre a integridade dos dados, principalmente agora que o Brasil tem a LGPD (Lei Geral de Proteção de Dados Pessoais), e você deve garantir que não haja vazamento de dados sensíveis. Vale ressaltar que, apesar de ser um sistema muito robusto na questão de segurança, o Linux não é imune a *malwares* e vírus. A implementação de práticas de segurança ajuda a proteger o servidor contra ameaças digitais que podem comprometer seu desempenho e segurança.

Este tópico exige conhecimento dos seguintes comandos:

- `find`
- `passwd`
- `fuser`
- `lsof`
- `nmap`
- `chage`
- `netstat`
- `sudo`
- `/etc/sudoers`
- `su`
- `usermod`
- `ulimit`
- `who` , `w` , `last`

13.1 Comando find

O comando **`find`** pode parecer ser apenas um comando simples de busca de arquivos ou diretórios, mas, com alguns ajustes, ele pode se tornar um belo aliado de administradores de sistemas que se preocupam com segurança, principalmente com permissões de arquivos. A seguir, vou demonstrar alguns exemplos de como usar o comando `find` para boas práticas de segurança em Linux.

Para encontrar arquivos que tenham permissões muito abertas (por exemplo, permissões que permitem a escrita para todos os usuários), você pode usar o comando `find` da seguinte forma:

```
# find /caminho/do/diretorio -type f -perm -o=w
```

Esse comando procura por *arquivos* que tenham permissões de escrita para outros usuários dentro do diretório especificado (`/caminho/do/diretorio`). Para encontrar *diretórios* com permissões

muito abertas, que permitam a escrita para todos os usuários, usamos este comando:

```
# find /caminho/do/diretorio -type d -perm -o=w
```

Outro ponto é a procura por arquivos pertencentes a um usuário específico. Neste caso, usamos:

```
# find /caminho/do/diretorio -type f -user nome_do_usuario
```

E para encontrar arquivos executáveis disponíveis a todos os usuários o comando é:

```
# find /caminho/do/diretorio -type f -perm -o=x
```

Recentemente, em uma empresa em que faço consultoria, alguns servidores foram alvo de uma invasão. O primeiro comando que executei foi o `find` para procurar por arquivos que foram modificados nos últimos dias; assim, eu já teria um "norte" para começar a vasculhar o que os invasores fizeram no ambiente. O comando que utilizei foi este, no qual inseri a instrução `mtime -7` para encontrar arquivos modificados nos últimos 7 dias:

```
# find /caminho/do/diretorio -type f -mtime -7
```

13.2 Comando passwd

O comando **`passwd`** no Linux é utilizado para alterar a senha de um usuário no sistema. Um administrador pode forçar um usuário a alterar sua senha na próxima vez que fizer login. Isso pode ser feito usando a opção `-e` com o comando `passwd`:

```
# passwd -e nome_do_usuario
```

Isso desativa a conta do usuário até que uma nova senha seja definida na próxima vez que o usuário fizer login. Para obter informações de status da conta, usamos `-s` ou `--status`. Por exemplo:

```
# passwd -S
juliano P 12/07/2023 0 99999 7 -1
```

Sendo:

- `P` : indica que o usuário possui uma senha válida (`P`); outros valores possíveis são `L` para uma senha bloqueada e `NP` para nenhuma senha;
- `12/07/2023` : data da última alteração da senha;
- `0` : idade mínima em dias (o número mínimo de dias entre mudanças de senha). Um valor de `0` significa que a senha pode ser alterada a qualquer momento;
- `99999` : idade máxima em dias (o número máximo de dias em que a senha é válida). Um valor de `99999` desabilita a expiração da senha;
- `7` : período de aviso em dias (o número de dias antes da expiração da senha em que um usuário será avisado);
- `-1` : período de inatividade da senha em dias (o número de dias inativos após a expiração da senha até que a conta seja bloqueada). Um valor de `-1` remove a inatividade de uma conta.

Bloqueando a senha de um usuário

Uma forma de bloquear e desbloquear a senha de um usuário é através do comando `usermod`. Para bloquear, utilize: `usermod -L juliano` ou `usermod --lock juliano`. Para desbloquear, `usermod -U juliano` ou `usermod --unlock juliano`.

Além dos comandos `passwd` e `usermod` para lidar com senhas, podemos usar o comando `chage` (de *Change age*, em português "alterar idade"), que veremos em breve ainda neste capítulo.

13.3 Comando fuser

O comando `fuser` fornece informações sobre quais processos estão utilizando determinados recursos, como arquivos ou diretórios. O `fuser` é útil para diagnosticar situações em que você deseja saber quais processos estão impedindo a liberação de um recurso, como um sistema de arquivos ou dispositivos.

Para exemplificar o uso do `fuser`, imagine que um administrador de sistemas montou o dispositivo `/mnt/c` para que usuários possuam acesso a arquivos de uma partição que contém o Windows. Quando você executa o comando para desmontar a partição, não consegue desmontá-la porque ela está em uso. Neste ponto, podemos usar o seguinte comando:

```
# fuser -m /mnt/c
/mnt/c:               352c   422c   423c
```

A saída do comando mostra alguns processos usando os dispositivos `352c`, `422c`, `423c`. Para saber de que usuários são esses processos, podemos usar o comando `fuser` da seguinte forma, onde `-m` significa *mount* e `-u`, *usuário*:

```
# fuser -mu /mnt/c
/mnt/c:               352c(juliano)   427c(root)   428c(root)
```

13.4 Comando lsof

O comando `lsof` (*List Open Files*) no Linux é utilizado para listar informações sobre os arquivos que estão sendo utilizados pelos processos em execução no sistema. Ele fornece uma visão detalhada dos arquivos abertos, *sockets* de rede, diretórios e outros recurso relacionados que estão sendo acessados pelos processos.

Por exemplo, para listar todos os arquivos abertos por um processo chamado `top`, você pode usar:

```
$ lsof -c top
COMMAND PID    USER   FD   TYPE DEVICE SIZE/OFF  NODE NAME
top     485 juliano  cwd    DIR   8,32     4096   477 /home/juliano
top     485 juliano  rtd    DIR   8,32     4096     2 /
top     485 juliano  txt    REG   8,32   133184  1897 /usr/bin/top
top     485 juliano  mem    REG   8,32    48152  6863 /usr/lib/x86_64-
linux-gnu/libnuma.so.1.0.0
top     485 juliano  mem    REG   8,32    27002 23528 /usr/lib/x86_64-
linux-gnu/gconv/gconv-modules.cache
top     485 juliano  mem    REG   8,32    86100 51005 /usr/share/locale-
langpack/pt_BR/LC_MESSAGES/procps-ng.mo
top     485 juliano  mem    REG   8,32  3053168 17374
/usr/lib/locale/locale-archive
top     485 juliano  mem    REG   8,32   149760  6731 /usr/lib/x86_64-
linux-gnu/libgpg-error.so.0.32.1
top     485 juliano  mem    REG   8,32  1296312  6711 /usr/lib/x86_64-
linux-gnu/libgcrypt.so.20.3.4
top     485 juliano  mem    REG   8,32    39024  6646 /usr/lib/x86_64-
linux-gnu/libcap.so.2.44
```

O `lsof` é uma ferramenta poderosa para diagnóstico e análise de sistemas, especialmente quando se trata de identificar quais processos estão interagindo com quais arquivos e recursos do sistema. O exame da certificação exige apenas este conhecimento superficial do comando.

13.5 Comando nmap

O comando **`nmap`** é uma abreviatura para *Network Mapper* e é uma ferramenta poderosa para explorar redes e realizar varreduras de segurança em hosts. A sintaxe básica do comando `nmap` é:

```
# nmap [opcoes] [alvo]
```

Exemplo de uso:

```
# nmap -p 80,443 localhost
Starting Nmap 7.80 ( https://nmap.org ) at 2023-12-27 09:35 -03
Nmap scan report for localhost (127.0.0.1)
Host is up (0.00011s latency).

PORT    STATE  SERVICE
80/tcp  closed http
443/tcp closed https

Nmap done: 1 IP address (1 host up) scanned in 0.02 seconds
```

O comando `nmap -p 80,443 localhost` vai varrer as portas 80 e 443 da sua máquina. No caso, vemos que as portas estão fechadas (`closed`). Você pode alterar `localhost` para o IP que deseja consultar.

Para verificar a versão do sistema operacional de um host, usamos:

```
# nmap -O 192.168.10.20
```

Vale ressaltar que, caso a máquina a ser pesquisada possua um firewall ativo, ele pode barrar ou ocultar as informações que você busca com o `nmap` .

Para uma varredura intensiva e detalhada, usamos:

```
# nmap -A 192.168.10.20
```

Com esse comando, o `nmap` tentará descobrir a versão do sistema operacional e todos os serviços em execução. Exemplo de saída do comando:

```
Starting Nmap 7.80 ( https://nmap.org ) at 2023-12-27 15:30 UTC
Nmap scan report for alvo.com (192.168.1.1)
Host is up (0.045s latency).
Not shown: 998 closed ports
PORT     STATE SERVICE    VERSION
22/tcp   open  ssh        OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux;
```

```
protocol 2.0)
80/tcp   open  http       Apache httpd 2.4.29 ((Ubuntu))
|_http-server-header: Apache/2.4.29 (Ubuntu)
|_http-title: Site Example
139/tcp  open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp  open  netbios-ssn Samba smbd 4.7.6-Ubuntu (workgroup: WORKGROUP)
...
Aggressive OS guesses: Linux 3.10 - 4.11, Linux 2.6.32 - 3.13, Linux
2.6.32 - 3.10, Linux 3.1, Linux 3.10 - 4.1, Linux 3.2 - 4.9
...
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Host script results:
|_clock-skew: mean: -2h00m00s, deviation: 0s, median: -2h00m00s
...
```

Nesse exemplo fictício, o `nmap` está varrendo um host chamado `alvo.com` com o endereço IP `192.168.1.1`. A saída inclui informações sobre os serviços em execução nas portas abertas, versões dos serviços, suposições sobre o sistema operacional e outras informações relevantes. Essa saída é uma visão geral simplificada; uma saída real do `nmap` seria mais detalhada e específica para o ambiente em questão.

13.6 Comando chage

O comando **`chage`** é usado para modificar as configurações relacionadas à expiração de senhas de usuários. Através dele configuramos a data de expiração, a idade mínima e máxima da senha, entre outras configurações relacionadas à segurança de senhas.

Para exibir informações sobre a expiração da senha de um usuário, execute:

```
# chage -l nome_do_usuario
```

Para definir uma data de expiração do usuário para, por exemplo, dia 1º de janeiro de 2023, usamos:

```
# chage -E 2023-01-01 juliano
```

Após executar o comando, a conta do usuário `juliano` expirará em 1º de janeiro de 2023. Para reativar esta conta após ela expirar, podemos ajustar uma data de expiração com um valor futuro ou usando a opção `-1`, que indica que a conta não tem data de expiração:

```
# chage -E -1 juliano
```

Outra ação que nós, administradores de sistemas Linux, realizamos é forçar a troca da senha do usuário no seu próximo login. Para isso, executamos:

```
# chage -d 0 juliano
```

Nesse código, `-d` é a opção para definir a data de expiração de senha, sendo `0` a indicação da data atual.

Para definir uma data mínima e máxima de expiração de senhas, usamos as opções `-m` para data mínima e `-M` para data máxima. Por exemplo: queremos que o usuário `juliano` mantenha sua senha por pelo menos 10 dias e deverá obrigatoriamente trocá-la após 30 dias. O comando seria:

```
# chage -m 10 -M 20 juliano
```

13.7 Comando netstat

O comando `netstat` fornece uma variedade de informações sobre conexões de rede, estatísticas de interface, tabelas de roteamento, *sockets* e outros detalhes relacionados à rede.

Para mostrar todas as conexões de nosso host, podemos usar o comando:

```
# netstat -a
```

Um exemplo da saída do comando:

```
Proto  Recv-Q  Send-Q  Endereço Local   Endereço Remoto    Estado
tcp         0       0  0.0.0.0:22       0.0.0.0:*          LISTEN
tcp         0       0  192.168.1.2:22   192.168.1.3:12345  ESTABLISHED
udp         0       0  0.0.0.0:68       0.0.0.0:*
```

Essa saída do comando significa que:

- Há uma conexão TCP na porta 22 (SSH) em modo de escuta (`LISTEN`);
- Outra conexão TCP está estabelecida na porta 22 entre o endereço local `192.168.1.2` e o endereço remoto 192.168.1.3, na porta `12345` ;
- Há uma conexão UDP na porta 68.

Explicação dos campos:

- **`Proto`** : o protocolo da conexão (`tcp` ou `udp`);
- **`Recv-Q`** **e** **`Send-Q`** : números de fila para recebimento e envio;
- **`Endereço Local`** : o endereço e porta do lado local da conexão;
- **`Endereço Remoto`** : o endereço e porta do lado remoto da conexão;
- **`Estado`** : o estado da conexão (por exemplo, `LISTEN` , `ESTABLISHED` etc.).

Para ver a tabela de roteamento, podemos usar a opção `-r` .
Observe um exemplo simplificado da saída:

```
# netstat -r
Tabela de Roteamento IP do Kernel
Destino      Roteador     MáscaraGen.     Opções Métrica Ref  Uso Iface
default      192.168.1.1  0.0.0.0         UG     0       0    0   eth0
192.168.1.0  *            255.255.255.0   U      0       0    0   eth0
```

Nessa saída, a primeira linha representa a rota padrão (`default`) para a qual o tráfego é roteado para o roteador `192.168.1.1` através da interface `eth0` . A segunda linha representa a rede local `192.168.1.0` , que é acessada diretamente através da interface `eth0` .

Explicação dos campos:

- **`Destino`** : a rede de destino;
- **`Roteador`** : o roteador para alcançar o destino;
- **`MáscaraGen.`** : a máscara de sub-rede;
- **`Opções`** : opções de roteamento;
- **`Métrica`** : a métrica associada à rota;
- **`Ref`** : número de referências à rota;
- **`Uso`** : número de pacotes que foram roteados por essa entrada;
- **`Iface`** : a interface de rede associada à rota.

As opções mais comuns exibidas pelo comando `netstat -r` são:

- **UG (Gateway Padrão):** indica a rota padrão, utilizada quando não há uma rota mais específica para um destino;
- **U (Up):** indica que a rota está ativa;
- **G (Gateway):** indica que há um roteador associado a essa rota;
- **H (Host):** indica que a rota é para um host específico, em oposição a uma rede;
- **! (Rejeitada)**: indica que a rota é inatingível.

13.8 O comando sudo e o arquivo /etc/sudoers

O comando **`sudo`** é uma abreviação de *superuser do*. Ele é utilizado para permitir que um usuário execute comandos com privilégios de superusuário ou de outro usuário, conforme configurado no arquivo **`/etc/sudoers`** .

Normalmente usamos o comando `sudo` para executar comandos administrativos. Exemplo:

```
$ sudo apt install vim
```

Esse comando serve para instalar o pacote `vim` e só é permitido com a conta *root*; entretanto, caso você utilize o comando `sudo`, durante a execução do comando seu usuário ganha poderes equivalentes a *root* e permite a instalação do programa. O comando `sudo` só fica disponível para usuários autorizados dentro do arquivo `/etc/sudoers`.

> **NOTA IMPORTANTE:** a edição do arquivo `/etc/sudoers` deve ser feita com muito cuidado, pois erros na sintaxe podem causar problemas sérios.

Aqui está um exemplo de como você pode adicionar a permissão total para o usuário `juliano`:

- Abra o arquivo `/etc/sudoers` para edição usando o comando `visudo`:

```
# visudo
```

- Adicione a linha `juliano ALL=(ALL:ALL) ALL` ao final do arquivo. Essa linha concede permissões totais para o usuário `juliano`. A explicação da sintaxe dessa linha é:
    - **`juliano`**: nome do usuário;
    - **`ALL=(ALL:ALL)`**: indica que `juliano` pode agir como qualquer usuário e em qualquer grupo;
    - **`ALL`**: permissão para executar qualquer comando.
- Salve e feche o editor (normalmente, no `visudo`, isso envolve pressionar Esc, digitar **`:wq`** e pressionar Enter).

O que pouca gente sabe é que é possível determinar quais comandos um usuário poderá executar como *root*. Desse modo,

você pode estabelecer privilégios diferentes para vários usuários. Por exemplo, quero que o usuário `juliano` tenha direito de instalar programas com o comando `apt`, entretanto ele não pode usar o `sudo` para nenhum outro comando, como para criação de usuários, configuração de rede etc. Para isso, execute `visudo` e adicione a seguinte linha:

```
juliano ALL=(ALL:ALL) /usr/bin/apt
```

Essa linha especifica que o usuário `juliano` pode executar o comando `/usr/bin/apt` como superusuário. Depois de fazer essas alterações, o usuário `juliano` será capaz de executar o comando `apt` com `sudo`. No entanto, ele não terá permissões totais e só poderá usar o `sudo` para executar o `apt`. Isso ajuda a restringir o escopo das operações que o usuário pode realizar com privilégios elevados.

13.9 Comando su e usermod

O comando **`su`** é utilizado para alternar usuários no terminal. Ele permite que um usuário entre em uma sessão de outro usuário sem fazer logout da sessão atual. O uso básico do comando `su` é o seguinte:

```
$ su usuário
```

Se você executar apenas `su` sem especificar um usuário, o comando assumirá que você deseja se tornar o superusuário (*root*).

Já o comando **`usermod`** é utilizado para modificar as configurações de um usuário no sistema. Para alterar, por exemplo, o grupo primário de um usuário e o seu diretório padrão, utilize:

```
# usermod -g novo_grupo -d /novo-diretorio usuario
```

Onde a opção para colocar um novo grupo é a `-g` e, para um novo diretório, a opção é `-d` .

Veja outras opções comuns do comando `usermod` :

- **`-c, --comment` COMENTÁRIO**: adiciona ou modifica o campo de comentário associado à conta do usuário. Pode incluir informações sobre o usuário. Exemplo:

```
usermod -c "John Doe" user1
```

- **`-d, --home` DIRETÓRIO**: define o diretório inicial do usuário. Especifica o caminho para o novo diretório `home` do usuário. Exemplo:

```
usermod -d /home/novo_user user1
```

- **`-g, --gid` GRUPO**: define o grupo primário do usuário para o grupo especificado. Exemplo:

```
usermod -g novo_grupo user1
```

- **`-aG, --append` GROUPS**: adiciona o usuário aos grupos especificados. Isso não remove o usuário de outros grupos. Exemplo:

```
usermod -aG grupo1,grupo2 user1
```

- **`-l, --login` NOVO_NOME**: renomeia a conta de usuário. Isso deve ser feito quando o usuário não está logado. Exemplo:

```
usermod -l novo_nome user1
```

- **`-s, --shell` SHELL**: modifica o shell padrão associado ao usuário. Exemplo:

```
usermod -s /bin/bash user1
```

- **`-u, --uid` UID**: define o ID do usuário. Pode ser usado para alterar o ID de usuário existente. Exemplo:

```
usermod -u 1001 user1
```

> **IMPORTANTE:** lembre-se de que algumas modificações podem exigir precauções adicionais, como garantir que o diretório `home` do usuário seja movido ou copiado corretamente para um diretório que tenha permissões adequadas. Certifique-se de fazer backup dos dados importantes antes de realizar alterações significativas em uma conta de usuário.

13.10 Os comandos ulimit, who, w e last

O comando **`ulimit`** é utilizado para controlar os limites de recursos do usuário no shell. Ele permite que você ajuste ou exiba os limites de recursos do sistema, como o número máximo de processos, o tamanho máximo do arquivo etc. Exemplo:

```
# ulimit -a
core file size          (blocks, -c) 0
data seg size           (kbytes, -d) unlimited
scheduling priority             (-e) 0
file size               (blocks, -f) unlimited
pending signals                 (-i) 256
max locked memory       (kbytes, -l) 16384
max memory size         (kbytes, -m) unlimited
open files                      (-n) 1024
pipe size            (512 bytes, -p) 8
POSIX message queues     (bytes, -q) 819200
real-time priority              (-r) 0
stack size              (kbytes, -s) 8192
cpu time               (seconds, -t) unlimited
max user processes              (-u) 256
virtual memory          (kbytes, -v) unlimited
file locks                      (-x) unlimited
```

Essa saída mostra os limites de recursos configurados para diferentes aspectos do ambiente do shell. Cada linha representa um limite específico e as unidades associadas. Note que os valores podem variar dependendo das configurações do sistema.

O comando `who` exibe informações sobre usuários que estão atualmente logados no sistema. Ele fornece detalhes como nome de usuário, terminal, data e hora de login, endereço IP, entre outros.

Exemplo:

```
# who
usuario1  tty1         2022-01-01 10:00 (:0)
usuario2  pts/0        2022-01-01 09:30 (192.168.1.100)
usuario3  pts/1        2022-01-01 08:45 (192.168.1.101)
```

Explicação da saída:

- `usuario1` está logado no terminal virtual `tty1` desde as 10:00 do dia 1º de janeiro de 2022;
- `usuario2` está logado na sessão de pseudo-terminal `pts/0` desde as 09:30 do dia 1º de janeiro de 2022 através do endereço IP `192.168.1.100`;
- `usuario3` está logado na sessão de pseudo-terminal `pts/1` desde as 08:45 do dia 1º de janeiro de 2022 através do endereço IP `192.168.1.101`.

A saída pode incluir informações adicionais, dependendo das opções específicas do comando e do ambiente do sistema. Essas informações são úteis para monitorar quem está conectado ao sistema, em que terminais e desde quando.

O comando `w` é usado para exibir informações detalhadas sobre os usuários que estão atualmente logados no sistema, incluindo detalhes sobre seus processos e atividades. Aqui está uma explicação do comando e um exemplo de saída:

```
# w
 10:00:01 up 5 days, 2:30, 3 users, load average: 0.05, 0.15, 0.10
```

```
USER     TTY    FROM           LOGIN@  IDLE   JCPU   PCPU WHAT
usuario1 tty1                  10:00   1:30m  0.10s  0.05s bash
usuario2 pts/0  192.168.1.100  09:30   2:00   0.20s  0.15s vim arquivo.txt
usuario3 pts/1  192.168.1.101  08:45   3:15   1:00   0.25s ssh servidor
```

Explicação da saída:

- Informações do sistema:
    - `10:00:01 up 5 days, 2:30` : indica o tempo atual, o tempo desde que o sistema foi inicializado (*uptime*), e o tempo médio de carga do sistema nos últimos 1, 5 e 15 minutos;
    - `3 users` : mostra o número total de usuários logados no sistema.
- Informações de usuários:
    - `USER` : nome do usuário;
    - `TTY` : terminal associado ao usuário;
    - `FROM` : indica de onde o usuário está conectado, seja um terminal local ou um endereço IP no caso de conexões remotas;
    - `LOGIN@` : hora de login;
    - `IDLE` : tempo que o usuário está ocioso;
    - `JCPU` : tempo total de CPU consumido pelo usuário desde o início da sessão;
    - `PCPU` : tempo total de CPU consumido pelo processo mais recente;
    - `WHAT` : Comando ou atividade atual do usuário.

O comando `w` é útil para administradores de sistemas monitorarem a atividade dos usuários, os terminais que estão utilizando e os processos em execução.

Para finalizar o capítulo, falarei sobre o comando `last` que é usado para exibir informações sobre as últimas entradas no arquivo de log de login do sistema. Ele mostra detalhes sobre as sessões de login dos usuários, incluindo o nome do usuário, o terminal de onde eles

fizeram login, o endereço IP (se aplicável), o início e o fim da sessão, entre outras informações.

Exemplo: Para mostrar as últimas sessões de login de um usuário específico, execute:

```
# last nome_do_usuario
```

Exemplo de saída de últimos usuários logados mostrando o endereço IP:

```
# last -i -a
usuario  pts/0    192.168.1.2    ter jan 10 09:30   ainda logado
usuario  tty1                     ter jan 10 08:45   ainda logado
outro    pts/1    192.168.1.3    ter jan 10 08:30   ainda logado
usuario  pts/0    192.168.1.2    ter jan 10 07:45 - 09:00  (01:15)
```

Conclusão

Qualquer administrador de sistemas Linux deve se preocupar com o tema segurança. O administrador é responsável pela prevenção de ameaças cibernéticas e pela proteção de dados sensíveis. Além disso, o bom administrador preocupa-se com a integridade dos sistemas e a garantia de disponibilidade do ambiente. Na prova da LPI, espera-se três perguntas sobre o tema.

Capítulo 14
Tópico 110.2 – Segurança do host

Certificação	LPIC-1
Versão	5.0
Tópico no exame	110.2 Segurança do host
Objetivo	Os candidatos devem saber como configurar um nível básico de segurança do computador.
Peso	3

Neste capítulo aprenderemos como melhorar a segurança de um host linux. O exame exige o básico sobre:

- Saber o que é senha sombreada (shadow) e como elas funcionam
- Desligar serviços de rede que não estão em uso
- Entender a função TCP Wrappers

Para iniciar, vamos explorar um pouco sobre a segurança de autenticação com senhas ocultas (shadow). Os componentes básicos dos dados da conta de um usuário são armazenados no arquivo `/etc/passwd` . Esse arquivo contém os campos:

- Nome de login
- Senha
- Identidade do usuário
- Identidade do grupo
- Comentário
- Localização do diretório inicial
- Shell padrão

Exemplo de uma linha de usuário no arquivo `/etc/passwd/` :

```
profjulianoramos:x:1000:1000:Prof. Juliano
Ramos,,,:/home/profjulianoramos:/bin/bash
```

Os campos são separados por dois pontos (`:`). Observe no exemplo acima que o segundo campo é representado pelo caractere `x` . Isso ocorre porque em sistemas linux modernos a senha não se encontra no arquivo `/etc/passwd/` e sim no arquivo `/etc/shadow` . Este arquivo não é legível para todos os usuários.

As configurações de senha são definidas pelos comandos `passwd` e `chage` . Ambos os comandos alteram a senha do usuário que está localizada em `/etc/shadow` . Como usuário `root` você pode definir a senha de um usuário através do comando:

```
# passwd profjulianoramos
```

Nesse exemplo, usei o comando `passwd` para alterar a senha do usuário `profjulianoramos` . Para obtermos informação sobre esta conta, usamos o comando `chage -l` :

```
# chage -l profjulianoramos
Última mudança de senha                    : jan 22, 2024
Senha expira                    : nunca
Senha inativa                    : nunca
Conta expira                       : nunca
Número mínimo de dias entre troca de senhas        : 0
Número máximo de dias entre troca de senhas        : 99999
Número de dias de avisos antes da expiração da senha    : 7
```

Para evitar que o usuário `profjulianoramos` faça um novo login no sistema, podemos definir uma senha anterior à data atual, neste caso, antes do dia `22/01/2024` . Logo, o comando seria:

```
# chage -E 2024-01-21 profjulianoramos
```

Uma alternativa seria usando o comando `chage -l` :

```
# chage -l profjulianoramos
```

Quando a conta está suspensa/bloqueada, ao tentar fazer login você será reportado com a informação:

```
# login profjulianoramos
Password:
Your account has expired; please contact your system administrator
Authentication failure
```

Analisando conexões de entrada de rede

Serviços como servidores web, servidores de e-mail e servidores de impressão operam por meio de portas dedicadas, executando-se simultaneamente. No sistema clássico, fundamentado em *Sys-V-init*, cada um desses serviços é controlado por meio do comando `service`. Contudo, em sistemas modernos baseados no `systemd`, adotamos o comando `systemctl` para a administração de serviços.

Antigamente, não era aconselhável executar vários serviços em portas dedicadas (também conhecidos como serviços autônomos) devido às limitações de recursos dos computadores. Para contornar essa restrição, utilizava-se um "superdaemon" para monitorar as conexões de rede de entrada e iniciar o serviço conforme necessário, mas isto tinha um lado negativo, a perda de desempenho. Entre esses "superdaemons" um dos mais conhecidos era o `inetd`.

O `inetd` é um daemon que iniciava e gerenciava serviços de rede sob demanda. Em vez de manter vários daemons de serviços de rede ativos o tempo todo, o `inetd` ficava à espera de conexões em portas específicas, quando uma solicitação de conexão era recebida em uma dessas portas, o `inetd` iniciava o serviço correspondente para lidar com a conexão. Nos sistemas atuais baseados em `systemd`, a unidade `systemd.socket` é usada de forma semelhante.

O `systemd.socket` é uma parte do sistema de inicialização `systemd` no Linux, que é usado para gerenciar sockets de rede, ele, permite a ativação sob demanda de serviços quando há atividade em um determinado socket, proporcionando uma abordagem mais eficiente e flexível para o gerenciamento de serviços de rede.

Ele funciona com a criação de um arquivo, geralmente com extensão `.socket`. Neste arquivo, você especifica detalhes como o tipo de socket (TCP ou UDP), a porta e outros parâmetros relevantes. Exemplo:

```
[Socket]
ListenStream=12345
Accept=true
```

- **Ativação sob demanda:** ao contrário de abordagens tradicionais, onde um serviço fica constantemente à espera de conexões em uma porta, o `systemd.socket` permite que um serviço seja iniciado apenas quando há atividade no socket.
- **Configuração do serviço associado:** além do arquivo de unidade do socket, é necessário configurar o serviço associado que será ativado quando uma conexão for detectada. O serviço é definido em um arquivo de unidade separado, geralmente com extensão `.service`.

Exemplo de um arquivo `meu_servico.service`:

```
[Service]
ExecStart=/caminho/do/meu/executavel
```

- **Relacionamento entre socket e serviço:** no arquivo de unidade do serviço, você especifica a relação com o socket correspondente, utilizando a opção Sockets.

Exemplo:

```
[Service]
Sockets=my_socket.socket
```

- **Habilitação e inicialização:** após criar os arquivos de unidade para o socket e o serviço, você habilita e inicia ambos.

```
#   systemctl enable meu_socket.socket
#   systemctl enable meu_servico.service
#   systemctl start meu_socket.socket
```

O `systemd` agora monitora o socket especificado e, quando há uma atividade, inicia o serviço associado.

Essa abordagem traz vantagens de eficiência, pois os serviços são iniciados somente quando necessário, economizando recursos do sistema. Além disso, facilita a configuração e o gerenciamento de serviços de rede, especialmente em ambientes dinâmicos onde as conexões podem ser intermitentes.

Buscando serviços desnecessários

Uma forma de controlar os recursos do sistema é monitorando os serviços em execução. Serviços desnecessários e não utilizados por uma questão de segurança e recurso devem ser desativados. Nos sistemas baseados em `SyS-V-Init` , podemos verificar o status de todos os serviços com o seguinte comando:

```
# service --status-all
```

Verifique a lista de saída e desative os serviços desnecessários usando:

```
# update-rc.d SERVICE-NAME remove
```

Em sistemas baseados em Red Hat, o comando é:

```
# chkconfig SERVICE-NAME off
```

Para sistemas modernos baseados em `systemd` , podemos usar o seguinte comando para listar todos os serviços em execução:

```
# systemctl list-units --state active --type service
```

Em seguida, desativamos as unidades desnecessárias com:

```
# systemctl disable UNIT --now
```

Esse comando interrompe o serviço e remove da lista de serviços, evitando que ele seja lançado na próxima inicialização do sistema. Além disso, para obter um levantamento dos serviços de rede que estão à escuta, podemos usar `netstat` em sistemas mais antigos (é necessário ter o pacote `net-tools` instalado).

```
# netstat -ltu
Conexões Internet Ativas (sem os servidores)
Proto Recv-Q Send-Q Endereço Local          Endereço Remoto         Estado
tcp        0      0 localhost:43753         0.0.0.0:*               OUÇA
tcp        0      0 localhost:43695         0.0.0.0:*               OUÇA
tcp        0      0 localhost:27060         0.0.0.0:*               OUÇA
tcp        0      0 localhost:57343         0.0.0.0:*               OUÇA
tcp        0      0 0.0.0.0:27036           0.0.0.0:*               OUÇA
tcp        0      0 localhost:ipp           0.0.0.0:*               OUÇA
tcp6       0      0 localhost:ipp           [::]:*                  OUÇA
udp        0      0 0.0.0.0:57913           0.0.0.0:*
udp        0      0 0.0.0.0:27036           0.0.0.0:*
udp        0      0 0.0.0.0:631             0.0.0.0:*
udp        0      0 mdns.mcast.net:mdns     0.0.0.0:*
udp        0      0 mdns.mcast.net:mdns     0.0.0.0:*
udp        0      0 0.0.0.0:mdns            0.0.0.0:*
udp6       0      0 [::]:58000              [::]:*
udp6       0      0 Certbest:dhcpv6-client  [::]:*
udp6       0      0 [::]:mdns               [::]:*
```

Em sistemas modernos, podemos usar o comando equivalente `ss` (serviços de socket):

```
# ss -ltu
Netid   State    Recv-Q   Send-Q                          Local
Address:Port                Peer Address:Port   Process
udp     UNCONN   0        0
0.0.0.0:57913                      0.0.0.0:*
udp     UNCONN   0        0
0.0.0.0:27036                      0.0.0.0:*
udp     UNCONN   0        0
0.0.0.0:631                        0.0.0.0:*
udp     UNCONN   0        0
```

```
224.0.0.251:mdns                    0.0.0.0:*
udp     UNCONN   0        0
224.0.0.251:mdns                    0.0.0.0:*
udp     UNCONN   0        0
0.0.0.0:mdns                      0.0.0.0:*
udp     UNCONN   0        0
[::]:58000                        [::]:*
udp     UNCONN   0        0
[fe80::8601:1469:9cef:8bac]%wlo1:dhcpv6-client              [::]:*
udp     UNCONN   0        0
[::]:mdns                         [::]:*
tcp     LISTEN   0        128
127.0.0.1:43753                     0.0.0.0:*
tcp     LISTEN   0        128
127.0.0.1:43695                     0.0.0.0:*
tcp     LISTEN   0        128
127.0.0.1:27060                     0.0.0.0:*
tcp     LISTEN   0        128
127.0.0.1:57343                     0.0.0.0:*
tcp     LISTEN   0        128
0.0.0.0:27036                     0.0.0.0:*
tcp     LISTEN   0        128
127.0.0.1:ipp                       0.0.0.0:*
tcp     LISTEN   0        128
[::1]:ipp                          [::]:*
```

TCP Wrappers

Nos tempos em que no linux não se tinha uma aplicação de firewall, TCP Wrappers eram usados para proteger as conexões de rede em um host. TCP Wrappers, também conhecido como (TCP Daemons) é uma tecnologia de controle de acesso em sistemas Unix-like que permite filtrar conexões de rede para serviços específicos. Eles atuam como uma camada de segurança adicional, permitindo ou negando o acesso a serviços de rede com base em regras definidas pelo administrador de sistema.

Nas distribuições recentes, os programas não obedecem mais a TCP Wrappers. Porém, para oferecer suporte aos sistemas linux legados

que ainda usam TCP Wrappers, vale a pena ter algum conhecimento básico sobre esta tecnologia específica. Seu conhecimento básico também é exigido no exame da LPI.

Como exemplo de seu funcionamento, usaremos o daemon SSH. O primeiro passo é verificar se ele usa a biblioteca `libwrap` que é o suporte a TCP Wrappers. Verificamos isso com o comando:

```
# ldd /usr/bin/sshd | grep "libwrap"
libwrap.so.0 => /lib/x86_64-linux-gnu/libwrap.so.0 (0x00007f91dbec0000)
```

A seguir, adicionamos a seguinte linha ao arquivo `/etc/hosts.deny`:

```
sshd: ALL
```

E por fim, adicionamos uma exceção no arquivo `/etc/hosts.allow` para conexões SSH da rede local:

```
sshd: LOCAL
```

As alterações não necessitam do reinício dos serviços. Nesse exemplo, estamos bloqueando conexões SSH externas e aceitando apenas as da rede local.

Conclusão

Apesar de ser um tópico pequeno para o exame de certificação, ele é de peso três, o que significa que podemos esperar até três questões sobre o assunto. Fique atento(a) principalmente com os comandos do `systemd` que são o futuro do sistema de gerenciamento do linux. Lembre-se sempre de realizar a leitura deste conteúdo à frente do seu computador, de preferência uma máquina virtual.

Capítulo 15
Tópico 110.3 – Proteção de dados com criptografia

Certificação	LPIC-1
Versão	5.0
Tópico no exame	110.3 Proteção de dados com criptografia
Objetivo	Os candidatos devem ser capazes de usar a criptografia de chave pública para proteger os dados e as comunicações.
Peso	4

A implementação de criptografia para proteção de dados é uma responsabilidade de extrema importância na gestão de sistemas, especialmente ao acessar sistemas remotamente. Em contraste com métodos inseguros como `telnet`, `rlogin` ou FTP, o protocolo SSH (Secure Shell) foi especificamente desenvolvido com foco em segurança. Utilizando criptografia de chave pública, o SSH autentica hosts e usuários, garantindo a segurança em todas as trocas de informações subsequentes. Adicionalmente, o SSH possibilita a criação de túneis de porta, permitindo que protocolos não criptografados transmitam dados através de uma conexão SSH segura. A versão recomendada atual do protocolo SSH é a 2.0, e o OpenSSH é uma implementação gratuita e de código aberto desse protocolo.

15.1 Configuração básica de um cliente

OpenSSH

Para se conectar a um host, execute o comando `ssh` especificando o usuário e o endereço IP ou nome de host:

```
# ssh profjuliano@192.168.10.30
The authenticity of host '192.168.10.30 (192.168.10.30)' can't be
established.
ECDSA key fingerprint is
SHA256:5JF7anupYipByCQm2BPvDHRVFJJixeslmppi2NwATYI.
Are you sure you want to continue connecting (yes/no)?
```

Confirme a conexão com `yes` . Como é a primeira vez que você estabelece uma conexão com este servidor remoto (192.168.10.30), sua autenticidade não pode ser verificada em nenhum banco de dados. Sendo assim, o servidor fornece uma `ECDSA key fingerprint` (Impressão de chave digital) de sua chave pública (usando a função hash `SHA256`).

Após a aceitação da conexão, a chave pública do servidor remoto foi incluída no banco de dados de hosts conhecidos, possibilitando a autenticação do servidor em conexões subsequentes. Essa lista de chaves públicas de hosts conhecidos é gerenciada no arquivo `known_hosts` , localizado em `~/.ssh` :

```
# exit
logout
Connection to 192.168.10.30 closed.
lucas@debian:~$ ls .ssh/
known_hosts
```

Tanto `.ssh` quanto `known_hosts` foram criados após a primeira conexão remota ter sido estabelecida. `~/.ssh` é o diretório padrão para configurações e informações de autenticação específicas do usuário.

Quando temos o mesmo nome de usuário no host a conectarmos com o ssh, não é preciso especificar o seu nome na conexão.

Exemplo:

```
profjuliano@debian:~$ ssh@192.168.10.30
```

Neste exemplo o usuário `profjuliano` local vai se conectar ao `profjuliano` remoto.

Agora, suponha que você queira estabelecer uma nova conexão remota com um host que por acaso tem o mesmo endereço IP de `profjuliano` (o que é comum se você usa DHCP em sua LAN). Aparecerá um aviso sobre a possibilidade de um ataque *man-in-the-middle*:

```
profjuliano@debian:~$ ssh barbara@192.168.10.30
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@ WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED! @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle
attack)!
It is also possible that a host key has just been changed.
The fingerprint for the ECDSA key sent by the remote host is
SHA256:KH4q3vP6C7e0SEjyG8Wlz9fVlf+jmWJ5139RBxBh3TY.
Please contact your system administrator.
Add correct host key in /home/profjuliano/.ssh/known_hosts to get rid of
this message.
Offending ECDSA key in /home/profjuliano/.ssh/known_hosts:1
  remove with:
 ssh-keygen -f "/home/profjuliano/.ssh/known_hosts" -R "192.168.10.30"
ECDSA host key for 192.168.1.77 has changed and you have requested strict
checking.
Host key verification failed.
```

Como não se trata de um ataque *man-in-the-middle*, você pode adicionar com segurança a impressão digital de chave pública do novo host a `.ssh/known_hosts`. Como a mensagem indica, o comando `ssh-keygen -f "/home/carol/.ssh/known_hosts" -R "192.168.10.30"` pode ser usado para remover a chave ofensiva (uma alternativa é usar `ssh-keygen -R 192.168.10.30` para remover todas as chaves

pertencentes a 192.168.10.30 de `~/.ssh/known_hosts`). Depois disso, será possível estabelecer uma conexão ao novo host.

15.2 Logins baseados em chave

Podemos configurar o cliente SSH para realizar o login sem a necessidade de fornecer uma senha, optando, em vez disso, pelo uso de chaves públicas. Essa abordagem é amplamente preferida para estabelecer conexões seguras com servidores remotos via SSH. O primeiro passo é gerar um par de chaves na máquina cliente, utilizando o comando `ssh-keygen` e a opção `-t` para especificar o tipo de criptografia desejado. No nosso caso, optamos pelo `Elliptic Curve Digital Signature Algorithm` (Algoritmo de Assinatura Digital de Curvas Elípticas). Em seguida, é necessário indicar o caminho para salvar o par de chaves (o diretório `~/.ssh/` é comum e o padrão recomendado) e fornecer uma frase secreta. Embora a frase secreta seja opcional, é altamente recomendável utilizá-la sempre.

Saída do comando:

```
# ssh-keygen -t ecdsa
Generating public/private ecdsa key pair.
Enter file in which to save the key (/home/profjuliano/.ssh/id_ecdsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/profjuliano/.ssh/id_ecdsa.
Your public key has been saved in /home/profjuliano/.ssh/id_ecdsa.pub.
The key fingerprint is:
SHA256:tlamD0SaTquPZYdNepwj8XN4xvqmHCbe8g5FKKUfMo8 carol@debian
The key's randomart image is:
+---[ECDSA 256]---+
| . |
| o . |
| = o o |
| B * |
```

```
| E B S o |
| o & O |
| @ ^ = |
| *.@ @. |
| o.o+B+o |
+----[SHA256]-----+
```

Ao criar o par de chaves, podemos passar a opção `-b` para `ssh-keygen` para especificar o tamanho da chave em bits (por exemplo: `ssh-keygen -t ecdsa -b 521`).

Agora, em seu diretório `~/.ssh` :

```
# ls
id_ecdsa id_ecdsa.pub known_hosts
```

`id_ecdsa` é a sua chave privada e `id_ecdsa.pub` é a sua chave pública.

Na criptografia assimétrica (também conhecida como criptografia de chave pública), as chaves pública e privada estão matematicamente relacionadas uma com a outra de tal forma que tudo o que é criptografado por uma só pode ser descriptografado pela outra.

A seguir, você precisa adicionar sua chave pública ao `arquivo ~/.ssh/authorized_keys` do usuário que vai se logar no host remoto (se o diretório `~/.ssh` ainda não existir, é preciso criá-lo primeiro). Há várias maneiras de copiar sua chave pública para o servidor remoto: usar uma unidade flash USB, através do comando `scp` , que transfere o arquivo usando SSH, ou ler (com `cat`) o conteúdo de sua chave pública e canalizá-lo para ssh desta maneira:

```
# cat id_ecdsa.pub |ssh ina@192.168.1.77 'cat >> .ssh/authorized_keys'
```

Uma vez que você adicionou a chave pública ao arquivo `authorized_keys` no host remoto, basta estabelecer uma conexão usando o usuário e o IP ou nome da máquina, não será necessário digitar senha. Entretanto, caso tenha colocado uma frase secreta, ela será necessária. A vantagem nesse caso é que a conexão depende de dois fatores:

- Chave privada SSH
- Frase secreta

15.3 O agente SSH

Existe uma abordagem que equilibra segurança e conveniência: empregar o agente de autenticação SSH (ssh-agent). Esse agente gera seu próprio shell e mantém em memória as chaves privadas, permitindo a autenticação por chave pública ao longo da sessão. Vamos examinar mais detalhadamente como ele opera:

1. Use `ssh-agent` para iniciar um novo shell bash:

   ```
   # ssh-agent /bin/bash
   ```

2. Use o comando `ssh-add` para adicionar sua chave privada a uma área da memória. Se tiver fornecido uma frase secreta ao gerar o par de chaves — o que é recomendado por razões de segurança — este será o momento de inseri-la:

```
# ssh-add
Enter passphrase for /home/profjuliano/.ssh/id_ecdsa:
Identity added: /home/profjuliano/.ssh/id_ecdsa (profjuliano@debian)
```

Após incluir sua identidade, é possível realizar o login em qualquer servidor remoto que tenha sua chave pública, dispensando a necessidade de inserir a senha novamente. Em sistemas modernos, é prática comum executar esse comando durante a inicialização do computador, uma vez que o agente permanecerá ativo na memória até que a máquina seja desligada (ou a chave seja descarregada manualmente).

O diretório de configuração global do SSH encontra-se no diretório: `/etc/ssh` :

```
# tree /etc/ssh
/etc/ssh
├── moduli
├── ssh_config
├── ssh_host_dsa_key
├── ssh_host_dsa_key.pub
├── ssh_host_ecdsa_key
├── ssh_host_ecdsa_key.pub
├── ssh_host_ed25519_key
├── ssh_host_ed25519_key.pub
├── ssh_host_rsa_key
├── ssh_host_rsa_key.pub
└── sshd_config
0 directories, 11 files
```

Os arquivos de configuração do cliente e do servidor SSH são respectivamente: `ssh.config` e `sshd.config`. A diferença entre `ssh_config` e `sshd_config` reside nos contextos em que são utilizados e nas configurações que cada um controla:

- `ssh_config`: este arquivo é utilizado no lado do cliente (cliente SSH). Pode incluir opções como identificação do usuário, chave privada a ser usada, entre outras configurações relacionadas ao comportamento do cliente SSH. Normalmente localizado em `/etc/ssh/ssh_config` para configurações globais e `~/.ssh/config` para configurações específicas do usuário.

- `sshd_config`: este arquivo é usado no lado do servidor (daemon SSH). Controla as configurações do servidor SSH (daemon). Inclui opções relacionadas à autenticação, permissões de usuário, portas de escuta, entre outras configurações de segurança e comportamento do servidor SSH. Geralmente é encontrado em `/etc/ssh/sshd_config`.

15.4 GPG

O GPG, ou *GNU Privacy Guard*, é uma implementação de software de código aberto do padrão OpenPGP (*Pretty Good Privacy*). Ele é uma ferramenta de criptografia e assinatura digital que fornece privacidade e autenticidade para comunicações de dados. Usamos o GPG para:

- Criptografia de mensagens
- Assinaturas Digitais
- Gerenciamento de chaves públicas e privadas
- Verificação de identidade

No Linux, usamos o GPG através da linha de comando.

O básico e essencial sobre GPG

O GPG possui diversos comandos e opções que podem ser utilizados para realizar operações de criptografia, descriptografia, assinatura digital, entre outras. Para criar um novo par de chaves criptográficas, usamos o comando:

```
# gpg --gen-key
```

Esse comando inicia o processo de geração de um novo par de chaves, onde você precisará fornecer informações como o tipo de chave e o tempo de validade. Para listar estas chaves públicas e privadas, usamos o comando:

```
gpg --list-keys
gpg --list-secret-keys
```

Para exportar uma chave pública de um determinado usuário para um arquivo, como `chave_publica.asc`, usamos o comando:

```
# gpg --export -a "Nome do Usuário" > chave_publica.asc
```

E para importar esta chave pública, usamos:

```
# gpg --import chave_publica.asc
```

Para criptografar um arquivo, usamos:

```
# gpg --encrypt --recipient "Nome do Destinatário" arquivo.txt
```

Neste comando, criptografamos o arquivo `arquivo.txt` usando a chave pública do "Nome do Destinatário". Para descriptografar, usamos:

```
# gpg --decrypt arquivo.txt.gpg
```

Descriptografa o arquivo criptografado `arquivo.txt.gpg` usando a chave privada correspondente.

Conclusão

Manter um servidor Linux seguro é de extrema importância devido a diversos motivos que impactam a integridade, confidencialidade e disponibilidade dos dados e serviços hospedados. O exame 102 da LPIC-1 não espera que você domine ferramentas de segurança avançadas, isto é exigido na certificação LPIC de nível 3, entretanto, estes tópicos básicos e fundamentais são essenciais para o Administrador de sistemas Junior.

Capítulo 16
Conclusões finais e próximos passos rumo à certificação

Este capítulo é crucial, pois marca um momento de reflexão profunda. Se você chegou até aqui, dedicando-se diante de um computador, completando os exercícios propostos e praticando os comandos repetidamente, quero lhe dar sinceros parabéns! Você está pronto(a) para marcar seu exame da LPI 102. No entanto, mesmo para quem não têm planos de realizar a prova agora, o conhecimento que você adquiriu com este conteúdo já o(a) eleva ao patamar de Administrador de Sistemas excepcional. Sinta-se orgulhoso(a) do seu progresso.

Ao longo de mais de quinze anos, minha trajetória tem sido dedicada aos servidores Linux, proporcionando-me a oportunidade de colaborar com profissionais de diversos níveis técnicos. Inegavelmente, quem mais se destacava entre eles eram meticulosos: aqueles que valorizavam a documentação e buscavam incessantemente aprimorar os processos. Em uma ocasião, recebi uma valiosa lição de um desses experts: "Juliano, nada é tão bom que não possa ser executado de maneira mais eficiente e econômica."

Essa frase deixou uma impressão duradoura em mim. Sempre que me envolvo no desenvolvimento de um programa, script ou processo, mantenho o hábito de revisá-los regularmente. Surpreendentemente, a cada revisão, descubro novas oportunidades para aprimorar o trabalho realizado.

Gostaria de estender esse mesmo princípio a este livro. Durante todo o processo de criação, fiz questão de revisar seu conteúdo meticulosamente antes de chegar às suas mãos. No entanto, reconheço que sempre há espaço para melhorias. Encorajo você a reler este livro, buscando maneiras de executar os mesmos

comandos de forma mais eficiente e econômica. Ficaria imensamente grato se compartilhasse suas sugestões e alterações por e-mail. Quem sabe não poderemos lançar uma próxima edição enriquecida com suas contribuições?

Caso opte por marcar o seu exame, tanto eu quanto a editora Casa do Código ficaríamos imensamente gratos em receber o seu valioso feedback. Desejamos a você muito sucesso em sua jornada em direção às certificações Linux, e esperamos ansiosamente encontrá-lo em breve para compartilhar um café e uma conversa agradável sobre o maravilhoso universo da tecnologia open source.

Agradecido,

Prof. Juliano Ramos - profjulianoramos@gmail.com